New methods for mode jumping in Markov chain Monte Carlo algorithms

Ibrahim, Adriana

Award date:
2009

Awarding institution:
University of Bath

New Methods for Mode Jumping in Markov Chain Monte Carlo Algorithms

submitted by

Adriana Irawati Nur Ibrahim

for the degree of Doctor of Philosophy

of the

University of Bath

Department of Mathematical Sciences

March 2009

Summary

Standard Markov chain Monte Carlo (MCMC) sampling methods can experience problem sampling from multi-modal distributions. A variety of sampling methods have been introduced to overcome this problem. The mode jumping method of Tjelmeland & Hegstad (2001) tries to find a mode and propose a value from that mode in each mode jumping attempt. This approach is inefficient in that the work needed to find each mode and model the distribution in a neighbourhood of the mode is carried out repeatedly during the sampling process. We shall propose a new mode jumping approach which retains features of the Tjelmeland & Hegstad (2001) method but differs in that it finds the modes in an initial search, then uses this information to jump between modes effectively in the sampling run. Although this approach does not allow a second chance to find modes in the sampling run, we can show that the overall probability of missing a mode in our approach is still low. We apply our methods to sample from distributions which have continuous variables, discrete variables, a mixture of discrete and continuous variables and variable dimension. We show that our methods work well in each case and in general, are better than the MCMC sampling methods commonly used in these cases and also, are better than the Tjelmeland & Hegstad (2001) method in particular.

Acknowledgements

I would like to thank:

My supervisor, Prof. Chris Jennison, for his guidance, help and support, as without his assistance completing this thesis would not have been possible;

The University of Malaya and the Malaysian government for supporting me financially;

My ex-officemates throughout the years — Steve G., Adam K., Mikhail C., Haojie Y., John S., Steve P. and Christos P. — for their help and support and for putting up with me all this while;

The staff in the Department of Mathematical Sciences and Faculty of Science Office, University of Bath, for their help;

My ex-housemates — especially Sawanya B. — and friends for their companionship;

And last but not least, my family, for just being *there* for me, though so far away.

Contents

Chapter 1

Introduction

Many statistical applications give rise to complex, high-dimensional distributions involving a great many variables and parameters. When the probability distributions involved are not analytically tractable, computational methods are needed to make inferences about the parameters in such models. In the case of Bayesian inference, we express prior knowledge probabilistically and combine this with the data we actually observe to obtain the "posterior" distribution of the unknown parameters. This posterior distribution encapsulates the information needed to draw inferences but learning about this distribution, even by simulation, can be a challenging task.

If sampling is possible from a complex, high-dimensional distribution, these samples can be used to learn about the shape of the distribution and reach decisions, draw inferences and make predictions. One of the major simulation methods is *Markov chain Monte Carlo* (MCMC) sampling. MCMC works by moving around the distribution and drawing samples while covering the distribution's support in the correct proportions. We shall state the general theory of Markov chains and describe how to construct an MCMC sampling algorithm in chapter 2.

We are interested in the problem of using MCMC to sample from multi-modal distributions. An example of multi-modality can be seen in image analysis, where different modes can appear depending on whether an object is present or absent in an image. MCMC sampling methods may face problems when sampling from multi-modal distributions as the Markov chain may not be able to move reliably between different modes. This is because to move between modes the chain has to go through parts with very low probability which, by definition, is something an MCMC algorithm is unlikely to do. We shall illustrate the problem of convergence when using MCMC sampling, especially when sampling from a multi-modal distribution, in chapter 2. Infrequent movement between modes will lead to incorrect proportions of time spent at each mode, while more frequent movement between modes will lead to better mixing of the

MCMC sampler and more reliable estimates.

In chapter 3 we review the special MCMC methods that have already been introduced to overcome the problem of moving between modes. Current mode jumping methods use special moves to jump between modes and, in the process they explore and sample from the distribution at the same time. The main example of this is the Tjelmeland & Hegstad (2001) method. We find this way of sampling to be inefficient and we propose a new, alternative mode jumping approach that learns more about the distribution initially by searching for the modes, and then uses the information about the modes to jump between them efficiently. We shall describe this new mode jumping approach in general in chapter 3. One way to search for the modes initially is by using an optimization method called "simulated annealing" (Kirkpatrick et al. (1983)), which we shall introduce in chapter 2. We shall discuss the potential efficiency gains from using our mode jumping approach in chapter 3, and provide evidence of this efficiency in an example with a continuous target distribution in chapter 4. However, there is an issue that the initial search may fail to find one of the modes. We shall address the theoretical aspect of this in chapter 3, where we show that we still get asymptotic theory for our approach as lengths of both the initial search and sampling run increase. In chapter 4 we show numerically how small the probability of failing to find a mode in the initial search can be.

Apart from applying our mode jumping method to the case where the target distribution has continuous variables (chapter 4), we shall also apply it to cases where the target distribution has discrete variables (chapter 5), a mixture of discrete and continuous variables (chapter 6), and variable dimension (chapter 7). We shall conclude the research reported in this thesis and discuss directions for future work in chapter 8.

Chapter 2

MCMC Sampling and Optimization via Simulated Annealing

2.1 MCMC Sampling

2.1.1 Markov chains for discrete variables

Definition of a Markov chain with discrete state space

Grimmett & Stirzaker (2001) state that a discrete-time stochastic process $X = \{X_0, X_1, X_2, \ldots\}$ is a Markov chain if it satisfies the *Markov property*

$$\mathbb{P}(X_t = s \,|\, X_0 = x_0, X_1 = x_1, ..., X_{t-1} = x_{t-1}) = \mathbb{P}(X_t = s \,|\, X_{t-1} = x_{t-1})$$

for all times $t \geq 1$ and $s \in S$, where S is the discrete state space. The evolution of a Markov chain is described by its '*transition probabilities*' $\mathbb{P}(X_t = s \,|\, X_{t-1} = x_{t-1})$. We shall be interested in chains that are time-homogenous; that is, the transition probabilities do not depend on time t. We define the transition matrix to be $\mathbf{P} = \{p_{ij}\}$, where

$$p_{ij} = \mathbb{P}(X_t = j \,|\, X_{t-1} = i).$$

Note that this definition holds even if X is multi-dimensional.

Theory of discrete Markov chains

A discrete Markov chain $\{X_t\}_{t=0}^{\infty}$ on S has a *stationary* (or invariant) distribution π if its transition matrix $\mathbf{P}$ satisfies the *general balance* equation

$$\pi \mathbf{P} = \pi, \tag{2.1.1}$$

which is to say

$$\sum_i \pi(i)p_{ij} = \pi(j) \quad \text{for all } j \in S.$$

Let $\mathbf{P}^t$ denote the product of t copies of $\mathbf{P}$. If X_0 has distribution π, the distribution of X_1 is $\pi\mathbf{P} = \pi$ and the distribution of X_t is $\pi\mathbf{P}^t = \pi$ for all $t \geq 0$. This explains why the distribution π satisfying equation (2.1.1) is called the stationary distribution for transition matrix $\mathbf{P}$.

To ensure that a Markov chain will have a stationary distribution π, its transition matrix $\mathbf{P}$ must satisfy equation (2.1.1). We are interested in cases where X is multi-dimensional or where the state space S is very large, so it may be very hard to create a $\mathbf{P}$ that will satisfy equation (2.1.1) for a given π. However, if a transition matrix $\mathbf{P}$ satisfies the stronger condition of *detailed balance*, that is

$$\pi(i)\, p_{ij} = \pi(j)\, p_{ji}, \qquad \text{for all } i, j \in S,$$

then that transition matrix also satisfies equation (2.1.1). To see this, we sum over i to obtain

$$\begin{aligned} \sum_i \pi(i)\, p_{ij} &= \sum_i \pi(j)\, p_{ji} \\ &= \pi(j) \sum_i p_{ji} \\ &= \pi(j). \end{aligned}$$

Detailed balance can be much easier to check directly. We shall see that detailed balance plays an important role in the construction of Markov chains with particular stationary distributions. Even though it is a stronger property than general balance, it is convenient to define a chain satisfying detailed balance for a given distribution π. A Markov chain that satisfies detailed balance is also called *reversible*.

Roberts (1996) states that for the distribution of $\{X_t\}$ to converge to a stationary distribution, the Markov chain needs to satisfy three important properties:

1. The chain has to be *irreducible*, i.e., from any starting point the chain can reach any state with some probability bigger than zero, in some finite number of iterations.

2. The chain needs to be *aperiodic*, i.e., it does not oscillate between different sets of states in a regular periodic movement.

3. The chain must be *positive recurrent*, i.e., for any initial state i, the probability of visiting a given state j at some time in the future is equal to 1 and $\mathbb{E}$(time of

the first return to state $i) < \infty$.

These three properties imply that there exists a unique stationary distribution π which satisfies general balance, i.e, $\pi\mathbf{P} = \pi$. If we run this Markov chain for a long time, the chain will gradually 'forget' its initial state and its probability distribution will converge to π, as stated by Roberts (1996) in the following theorem:

Theorem 2.1 *If X is irreducible, positive recurrent and aperiodic, then it has a stationary distribution π which is also its limiting distribution. We then say X is ergodic and*

1. $\mathbb{P}(X_t = s \,|\, X_0 = x_0) \to \pi(s)$ *as* $t \to \infty$ *for all* s *and all* x_0.
2. *For any function* f,

$$\frac{1}{t}\sum_{i=1}^{t} f(x_i) \to \mathbb{E}_\pi(f(X)) \text{ almost surely as } t \to \infty.$$

The main ideas of Markov chains stated for the discrete case can be extended to general state spaces (see Tierney (1996)). We shall state the equivalent ideas for the case of a continuous state space in the next section.

2.1.2 Markov chains for continuous variables

Definition of a Markov chain with continuous state space

Suppose X_t can take values in a continuous state space S in $\mathbb{R}^h$. The process $X = \{X_0, X_1, X_2, \ldots\}$ is a Markov chain if it satisfies the Markov property

$$\mathbb{P}(X_t \in A \,|\, X_0, X_1, ..., X_{t-1}) = \mathbb{P}(X_t \in A \,|\, X_{t-1})$$

for all $t \geq 1$ and any set $A \subset S$. In this case we consider the transition probabilities in the form of the transition kernel P. If the current state is x, $P(x, y)$ defines the probability density of moving to the value y. For now we shall think of P as an h-dimensional density.

Theory of continuous Markov chains

A continuous state Markov chain $\{X_t\}_{t=0}^{\infty}$ on S has a stationary distribution π if its transition kernel P satisfies the general balance equation

$$\int\int_A \pi(x)P(x, y)\,\mathrm{d}y\,\mathrm{d}x = \pi(A), \tag{2.1.2}$$

for all sets $A \subset S$. A continuous state Markov chain satisfies detailed balance and in turn, general balance, if

$$\int_{x\in A}\int_{y\in B} \pi(x)P(x,y)\,\mathrm{d}y\,\mathrm{d}x = \int_{y\in B}\int_{x\in A} \pi(y)P(y,x)\,\mathrm{d}x\,\mathrm{d}y \tag{2.1.3}$$

for all A, $B \subset S$.

Tierney (1996) states that for general state-space Markov chains (which includes continuous ones), irreducibility is defined with respect to a distribution:

Definition 2.1 *Let the first return time of a Markov chain to a set $A \subset S$ be denoted by τ_A, i.e.,*

$$\tau_A = \inf\{t \geq 1 : X_t \in A\}.$$

$\tau_A = \infty$ means that the chain never returns to A. Then a Markov chain is ν-irreducible for a probability distribution ν on S if $\nu(A) > 0$ for a set $A \subset S$ implies that

$$P(\tau_A < \infty | X_0 = x) > 0$$

for all $x \in S$. A chain is irreducible if it is ν-irreducible for some probability distribution ν.

Tierney (1996) also states that if $\{X_t\}_{t=0}^{\infty}$ is irreducible and $\pi P = \pi$, then this chain is π-irreducible and positive recurrent, and π is the unique stationary distribution of the chain.

Let $P^t(x, A) = P(X_t \in A | X_0 = x)$. Also let $||\nu_1 - \nu_2||$ denotes the total variation distance between two probability distributions ν_1 and ν_2 on the same state space S, defined as

$$||\nu_1 - \nu_2|| = \sup_{A\subset S} |\nu_1(A) - \nu_2(A)|.$$

For a probability distribution ν on S, let a statement holds for 'ν-almost all x' if ν gives probability zero to the set of points in S where the statement fails. Then the ergodic theorem for general state-space Markov chains is stated by Tierney (1996) as:

Theorem 2.2 *Suppose $\{X_t\}$ is an irreducible, aperiodic Markov chain with transition kernel P and stationary distribution π. Then*

1. *The limiting distribution of the chain is the stationary distribution regardless of the starting values of the chain, i.e.,*

$$||P^t(x_0, \cdot) - \pi(\cdot)|| \to 0 \quad \text{as} \quad t \to \infty$$

for π-almost all x_0.

2. *For any real-valued function f,* $\mathbb{P}\left(\frac{1}{t+1}\sum_{i=0}^{t} f(X_i) \rightarrow \mathbb{E}_\pi(f(X)|X_0 = x_0)\right) = 1$ *for π-almost all x_0.*

2.1.3 Construction of MCMC sampling algorithms

To sample from a target distribution π using MCMC sampling, we construct a Markov chain that has a stationary distribution which is precisely our distribution of interest π. To do this, it will suffice to construct a Markov chain satisfying detailed balance with respect to the desired stationary distribution π. One method for doing this will be described below.

The Metropolis-Hastings Algorithm

Metropolis et al. (1953) first suggested an algorithm which was later on generalized by Hastings (1970), and the result is called the *Metropolis-Hastings* algorithm. Denote the state at time t by X_t. If $X_t = x$, a proposal Y is generated from a proposal distribution $q(x, y)$, defined as a discrete distribution for y in the discrete case or a density for y in the continuous case. The acceptance probability for this proposal is defined to be

$$\alpha(x, y) = \min\left\{1, \frac{\pi(y)\, q(y, x)}{\pi(x)\, q(x, y)}\right\}. \tag{2.1.4}$$

With probability $\alpha(x, y)$ we set $X_{t+1} = y$, otherwise we set $X_{t+1} = x$. In the discrete case, this process defines a transition matrix $\mathbf{P} = (p_{ij})$, which we now show satisfies detailed balance. For any pair of states x and y in S with $y \neq x$,

$$\begin{aligned}
\pi(x)\, p_{xy} &= \pi(x)\, q(x, y)\, \alpha(x, y) \\
&= \pi(x)\, q(x, y) \min\left\{1, \frac{\pi(y)\, q(y, x)}{\pi(x)\, q(x, y)}\right\}.
\end{aligned}$$

Considering the two cases where $\pi(y)\, q(y, x) > \pi(x)\, q(x, y)$ and vice versa, we see that

$$\pi(x)\, q(x, y) \min\left\{1, \frac{\pi(y)\, q(y, x)}{\pi(x)\, q(x, y)}\right\} = \pi(y)\, q(y, x) \min\left\{1, \frac{\pi(x)\, q(x, y)}{\pi(y)\, q(y, x)}\right\}$$

and, therefore,

$$\begin{aligned}
\pi(x)\, p_{xy} &= \pi(y)\, q(y, x) \min\left\{1, \frac{\pi(x)\, q(x, y)}{\pi(y)\, q(y, x)}\right\} \\
&= \pi(y)\, q(y, x)\, \alpha(y, x) \\
&= \pi(y)\, p_{yx}.
\end{aligned}$$

For any pair of states x and y in S with $y = x$ it is true trivially that $\pi(x)\, p_{xy} = \pi(y)\, p_{yx}$.

For the continuous case, this process will define a transition kernel $P(x, y)$ for states

$y \neq x$ that can be shown to satisfy detailed balance in a similar way as the discrete case. However, in this case if $y = x$, $P(x, y)$ is a point mass probability of rejecting all possible proposals Y instead of a probability density.

In this algorithm, the proposal distribution q can take any form as long as the transition kernel or matrix is π-irreducible and aperiodic. For the discrete case, we know that the transition kernel or matrix is aperiodic as long as the probability of staying in state x is greater than zero for some state x. This normally happens in the Metropolis-Hastings algorithm since values of $\alpha(x, y) < 1$ imply the probability of staying in state x is greater than zero for many x. For the continuous case, the same principle also applies, considering the discrete probability of rejecting all possible proposals Y. We can also apply this method even though we only know the target distribution up to a normalizing constant, i.e., we know $\pi(x) = c\,\psi(x)$ where $\psi(x)$ is known but c is unknown: the value of c cancels in the ratio of $\pi(x)$ and $\pi(y)$ in $\alpha(x, y)$. Note that in the original Metropolis et al. (1953) algorithm, the proposal distribution q is chosen to be symmetric, i.e., $q(x, y) = q(y, x)$, so that

$$\alpha(x, y) = \min\left\{1, \frac{\pi(y)}{\pi(x)}\right\}.$$

Hastings (1970) then generalized this method by proposing to use a general q instead.

Metropolis et al. (1953) originally proposed this algorithm as a single component sampler. If X can be split into components $(X^{(1)}, \ldots, X^{(h)})$, then each of these components can be updated one by one. This method is called the *single-component* Metropolis-Hastings. An update of a single component $X^{(i)}$ is described below. Suppose the current state is $X = x = (x^{(1)}, \ldots, x^{(h)})$. The next state X' is chosen by sampling a proposal $Y = y = (x^{(1)}, \ldots, x^{(i-1)}, y^{(i)}, x^{(i+1)}, \ldots, x^{(h)})$ from a proposal distribution $q_i(x, y)$. We calculate the acceptance probability $\alpha_i(x, y)$, where

$$\alpha_i(x, y) = \min\left\{1, \frac{\pi(y)\, q_i(y, x)}{\pi(x)\, q_i(x, y)}\right\}. \tag{2.1.5}$$

With probability $\alpha_i(x, y)$ we set $X' = y$, otherwise we set $X' = x$. In the discrete case, this process defines a transition matrix $\mathbf{P}_i$ which satisfies detailed balance. Then for a cycle of updates on elements 1 to h we define the overall transition matrix $\mathbf{P}$ as $\mathbf{P} = \mathbf{P}_1\mathbf{P}_2 \ldots \mathbf{P}_h$. At each time t, from current state X_t we generate the next state X_{t+1} by applying transition matrices $\mathbf{P}_1 \ldots \mathbf{P}_h$ in turn, which is equivalent to applying the overall transition matrix $\mathbf{P}$. Since $\mathbf{P}_i$ satisfies detailed balance, it also satisfies general balance, i.e., $\pi\mathbf{P}_i = \pi$, hence

$$\begin{aligned}\pi\mathbf{P} &= \pi\mathbf{P}_1\mathbf{P}_2\ldots\mathbf{P}_h \\ &= \pi\mathbf{P}_2\ldots\mathbf{P}_h \\ &\vdots \\ &= \pi,\end{aligned}$$

and we see that $\mathbf{P}$ satisfies general balance. This means that the single-component Metropolis-Hastings algorithm will produce a Markov chain with π as its stationary distribution.

In the continuous case, an update of a single component $X^{(i)}$ using the single-component Metropolis-Hastings will define a transition kernel P_i that does not use the full h-dimensional space. This is because for a given x, $q_i(x, y)$ takes us to a 1-dimensional subspace of S. However, the transition kernel P_i still satisfies detailed balance, i.e., equation (2.1.3). To see this let $y = (x^{(1)}, \ldots, x^{(i-1)}, y^{(i)}, x^{(i+1)}, \ldots, x^{(h)})$, then we need to show

$$\int_{x\in A}\int_{y^{(i)}:\, y\in B} \pi(x)\, q_i(x,y)\, \alpha_i(x,y)\, \mathrm{d}y^{(i)}\, \mathrm{d}x = \int_{y\in B}\int_{x^{(i)}:\, x\in A} \pi(y)\, q_i(y,x)\, \alpha_i(y,x)\, \mathrm{d}x^{(i)}\, \mathrm{d}y.$$

This equation can be expanded to become

$$\begin{aligned}&\int_{x^{(-i)}:\, x\in A}\int_{x^{(i)}:\, x\in A}\int_{y^{(i)}:\, y\in B} \pi(x)\, q_i(x,y)\, \alpha_i(x,y)\, \mathrm{d}y^{(i)}\, \mathrm{d}x^{(i)}\, \mathrm{d}x^{(-i)} \\ &= \int_{y^{(-i)}:\, y\in B}\int_{y^{(i)}:\, y\in B}\int_{x^{(i)}:\, x\in A} \pi(y)\, q_i(y,x)\, \alpha_i(y,x)\, \mathrm{d}x^{(i)}\, \mathrm{d}y^{(i)}\, \mathrm{d}y^{(-i)},\end{aligned}$$

and we can see that the left-hand side equals to the right-hand side since $x^{(-i)}$ and $y^{(-i)}$ take the same sets of values for $x \in A$ and $y \in B$ respectively, and the integrands are equal when $x^{(-i)} = y^{(-i)}$ by definition of α_i. If the transition kernel P_i satisfies detailed balance then it also satisfies general balance.

For computing efficiency, rather than calculating $\pi(y)/\pi(x)$ by finding $\pi(x)$ and $\pi(x)$, we can calculate

$$\frac{\pi_{X^{(i)}|X^{(-i)}}(y^{(i)}|x^{(-i)})}{\pi_{X^{(i)}|X^{(-i)}}(x^{(i)}|x^{(-i)})}$$

instead, where $X^{(-i)} = (X^{(1)}, \ldots, X^{(i-1)}, X^{(i+1)}, \ldots, X^{(h)})$ and $\pi_{X^{(i)}|X^{(-i)}}(x^{(i)}|x^{(-i)})$ is the full conditional distribution for $X^{(i)}$ under π, i.e., it is the conditional distribution of the i^{th} component of X given all the remaining components $X^{(-i)}$. This is because for $y = (x^{(1)}, \ldots, x^{(i-1)}, y^{(i)}, x^{(i+1)}, \ldots, x^{(h)})$,

$$\begin{aligned}\frac{\pi(y)}{\pi(x)} &= \frac{\pi_{X^{(i)}|X^{(-i)}}(y^{(i)}|x^{(-i)}) \times \pi_{X^{(i)}}(x^{(-i)})}{\pi_{X^{(i)}|X^{(-i)}}(x^{(i)}|x^{(-i)}) \times \pi_{X^{(i)}}(x^{(-i)})} \\ &= \frac{\pi_{X^{(i)}|X^{(-i)}}(y^{(i)}|x^{(-i)})}{\pi_{X^{(i)}|X^{(-i)}}(x^{(i)}|x^{(-i)})}. \end{aligned} \tag{2.1.6}$$

One special case of the single-component Metropolis-Hastings is the *Gibbs sampler* (Geman & Geman (1984)). For the Gibbs sampler, the proposal distribution for the i^{th} component is the full conditional distribution for $X^{(i)}$,

$$q_i(x, y) = \pi_{X^{(i)}|X^{(-i)}}(y^{(i)}|x^{(-i)}). \tag{2.1.7}$$

If we substitute equations (2.1.7) and (2.1.6) into equation (2.1.5), we get an acceptance probability of 1. This means that the Gibbs sampler proposal will always be accepted.

2.1.4 Using a variety of move types

Suppose that when using MCMC to sample from π, we want to use multiple "types" of moves. We denote a move type as ϕ, where $\phi \in \Phi$, with transition kernel P_ϕ for move type ϕ. If each P_ϕ satisfies general balance, then $\pi P_\phi = \pi$. We can select the move types at random, i.e., ϕ is generated from a distribution $f(\phi)$. Here, the overall transition matrix is $P = \int_\Phi f(\phi) P_\phi \, \mathrm{d}\phi$. This overall transition matrix satisfies general balance with respect to π as

$$\begin{aligned}\int_S \pi(x) P(x, y) \, \mathrm{d}x &= \int_S \pi(x) \int_\Phi f(\phi) P_\phi(x, y) \, \mathrm{d}\phi \, \mathrm{d}x \\ &= \int_\Phi f(\phi) \int_S \pi(x) P_\phi(x, y) \, \mathrm{d}x \, \mathrm{d}\phi \\ &= \int_\Phi f(\phi) \pi(y) \, \mathrm{d}\phi \\ &= \pi(y).\end{aligned}$$

This means that a MCMC sampling algorithm that uses several types of moves which are chosen at random, where the transition matrix for each move type satisfies detailed balance with respect to π, will produce a Markov chain that has π as its stationary distribution.

One may also select the move types using a pre-fixed cycle of move types. Using a pre-fixed cycle of move types will still generate an overall transition kernel P that satisfy general balance with respect to π, just as we saw for the single-component Metropolis-Hastings algorithm. The general theory then applies to the sequence of states formed by these "overall" transitions.

2.1.5 Rate of convergence

Roberts (1996) states that a reversible discrete Markov chain X, where reversible means that its transition matrix satisfies detailed balance, is geometrically ergodic in total variation distance if it is ergodic (positive recurrent and aperiodic) and if there exist $0 \leq \lambda < 1$ and a function $V(\cdot) > 1$ such that

$$\sum_j |\mathbb{P}(X_t = j | X_0 = i) - \pi(j)| \leq V(i)\lambda^t \tag{2.1.8}$$

for all i. The smallest λ for which there exists a function V satisfying (2.1.8) is called the *rate of convergence*, and is denoted by λ^*. Let $\{\lambda_0, \lambda_1, \ldots\}$ be the set of eigenvalues of the transition matrix $\mathbf{P}$, where $\lambda_0 = 1$ and $|\,\lambda_k\,| < 1$ for $k > 0$. Roberts (1996) states an equivalent definition for λ^*:

$$\lambda^* = \sup_{k>0} |\,\lambda_k\,|.$$

Note that we can use the value of λ^* to gauge the overall convergence of a discrete Markov chain; the smaller the value of λ^* is, the faster the distribution of the chain will converge to its equilibrium distribution.

2.1.6 Estimation

After sampling from our target distribution π using MCMC sampling methods, we can use the values to estimate $\mathbb{E}_\pi(f(X))$, the expectation of a function f under π. One example of such a function is the indicator function of an event $A \subset S$, $I\{x \in A\}$, for which $\mathbb{E}_\pi(I\{X \in A\}) = \mathbb{P}_\pi(A)$. Then we can estimate the probability of event A by estimating $\mathbb{E}_\pi(I\{X \in A\})$.

In using samples from a Markov chain with stationary distribution π to estimate the properties of π, it is advisable to discard values from the initial part of the chain, referred to as the *burn-in* period. This is because these burn-in values may not represent the values from the stationary distribution π, especially if the chain started in a region unlikely to be observed under π. If a burn-in period of length m is used in a chain of length n, we have the estimator

$$\bar{f}_n = \frac{1}{n-m} \sum_{t=m+1}^{n} f(x_t). \tag{2.1.9}$$

Convergence of $\bar{f}_n$ to $\mathbb{E}_\pi(f(X))$ is ensured in the discrete case by Theorem 2.1 and in the continuous case by Theorem 2.2.

We can measure how well our estimator $\bar{f}_n$ can estimate $\mathbb{E}_\pi(f(X))$ by looking at

the variance of $\bar{f}_n$. Green & Han (1992) derive the variance of $\bar{f}_n$, for large n, to be

$$\begin{aligned} \mathrm{Var}(\bar{f}_n) &= \frac{1}{n^2}\sum_{s=1}^{n}\sum_{t=1}^{n}\mathrm{cov}(f(x_s), f(x_t)) \\ &\simeq \frac{\sigma^2}{n}\sum_{t=-\infty}^{\infty}\rho_t(f) \end{aligned} \tag{2.1.10}$$

where σ^2 is the equilibrium variance of $f(x)$ and $\rho_t(f)$ is the autocorrelation function of the process $\{f(X_t)\}$ calculated under the equilibrium distribution π. They then define the integrated autocorrelation time (IAC) of the function f, $\tau(f)$, to be

$$\tau(f) = \sum_{t=-\infty}^{\infty}\rho_t(f).$$

This means that the value of $\mathrm{Var}(\bar{f}_n)$ depends on the value of $\tau(f)$, as σ^2 is fixed under equilibrium; a small value of $\tau(f)$ would indicate a good estimation performance. Note that if the samples are independent and identically distributed, then $\tau(f)$ would be equal to 1 as in this case $\mathrm{Var}(\bar{f}_n) \simeq \sigma^2/n$. Therefore, for large n, the value of $\mathrm{Var}(\bar{f}_n)$ estimated using correlated samples of size n from the Markov chain is equal to the value of $\mathrm{Var}(\bar{f}_n)$ estimated using independent and identically distributed samples of size $n/\tau(f)$. Green & Han (1992) note that in the discrete case, good asymptotic mean squared error of estimation can be helped by having a transition matrix $\mathbf{P}$ that has negative eigenvalues.

2.1.7 Problem of convergence when using MCMC sampling

One important thing to note is that the performance of our estimation of the expectation of f depends on the fact that our samples $\{X_1, \ldots, X_n\}$ are dependent samples approximately from π, i.e., the distribution of our simulated Markov chain has already converged to π. If the distribution of the simulated chain converges slowly to π, then the observed $\bar{f}_n$ may not be close to the required expectation. One reason why the distribution of the simulated chain may have problems converging to π is that the chain may have problems moving around the state space S, which can happen in practice when applying MCMC to high-dimensional problems, especially when the distributions are multi-modal. A chain that cannot move well between the modes of a multi-modal distribution will mix poorly, resulting in unrepresentative samples and slow convergence of the distribution of the chain to the target distribution. We shall illustrate the problem of using MCMC to sample from a multi-modal distribution in a simple example.

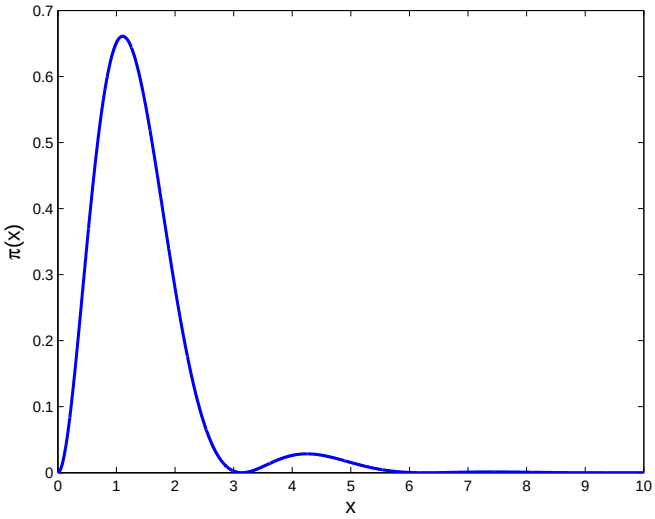


Figure 2-1: The function $\pi(x) = k \sin^2(x) \exp(-x)$, where $k = \{\int_0^\infty \sin^2(x) \exp(-x)\, \mathrm{d}x\}^{-1}$.

Example 1

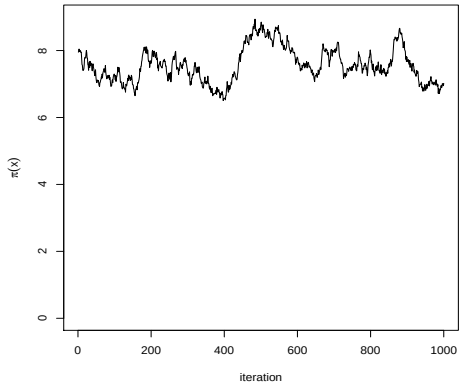
Suppose we want to sample from the distribution

$$\pi(x) = k \sin^2(x) \exp(-x), \qquad x \geq 0. \tag{2.1.11}$$

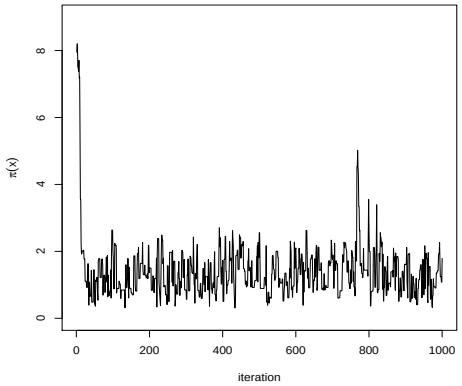
A plot of the function $\pi(x)$ is shown in Figure 2-1. This distribution has more than one mode: the global maximum is at $x = 1.1$, and there is a local maximum at $x = 4.2$. We shall sample from this distribution using the Metropolis-Hastings algorithm. We use a normal proposal distribution, taking $q(x, y)$ to be the density of the $N(x, \delta^2)$ distribution. For comparison purposes we shall run this algorithm with different values of δ, namely $\delta = 0.1$, 1.0, 2.0, 3.0, 4.0 and 5.0. We use the same starting point for all the simulated chains.

An informal way to check on convergence properties is to examine a time series plot of the simulations. The time series plots of the MCMC sampling of $\pi(x)$ for $n = 1{,}000$ iterations are shown in Figure 2-2. From this, we can see that the mixing rate varies greatly with the value of δ that we use. Smaller values of δ mean it is more difficult for the chain to jump between the modes. In particular, note that when sampling using $\delta = 0.1$ the chain is stuck in a minor mode for the whole run. However, if the value of δ is too big there is less movement in the chain as proposals to a state with very low probability become more frequent, and keep getting rejected.

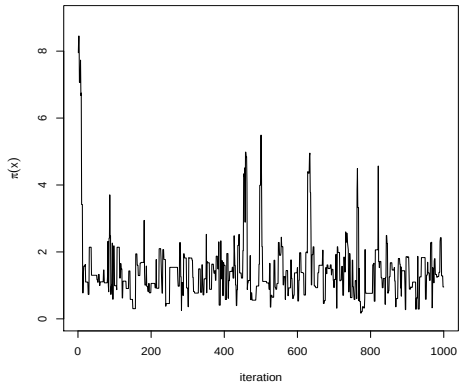
Gelman (1996) suggested that we can monitor convergence by comparing several sequences of MCMC simulations drawn from different starting points, and checking



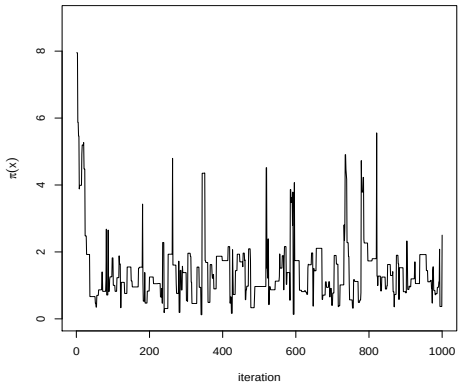
(a) $\delta = 0.1$



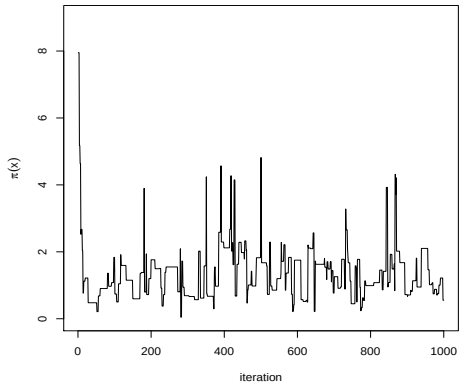
(b) $\delta = 1.0$



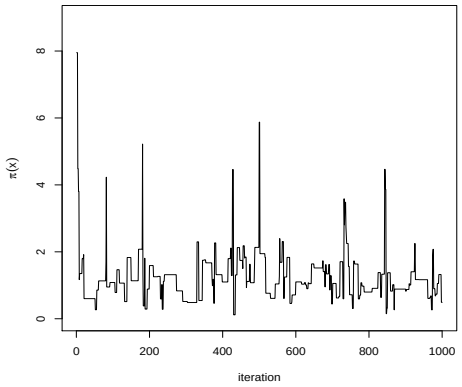
(c) $\delta = 2.0$



(d) $\delta = 3.0$



(e) $\delta = 4.0$



(f) $\delta = 5.0$

Figure 2-2: Time-series plots of the MCMC sampling of $\pi(x)$ for the various values of δ, with a run length of n = 1000.

that they are similar. One way of checking whether these sequences are similar is by looking at overlaid time series plots. We simulated from the target distribution $\pi(x)$ using five parallel sequences, with five different chosen starting points 2.5, 3.5, 9.0, 10.0 and 14.0. The overlapping time series plots resulting from five parallel sequences of length $n = 1,000$ for the various value of δ are shown in Figure 2-3. From Figure 2-3, we can see that the chains simulated by using $\delta = 0.1$ are mixing very slowly.

This example shows that the choice of proposal distribution plays a very important part when using a typical MCMC sampling method such as the Metropolis-Hastings algorithm, as it influences the mixing rate of the Markov chain. This is because in multi-modal target distributions, such as the one in this example, the mixing rate is influenced by the ability of the chain to move between modes. In this example, the problem of jumping between modes is easily solved by adjusting the value of δ. However, this problem might be harder to solve in other more complicated applications. Our motivation for the research reported in this thesis is that we want to find methods that can produce Markov chains with more frequent jumps between modes.

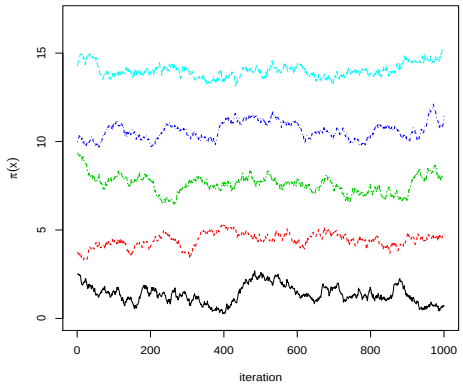
2.2 Optimization via simulated annealing

In the method we shall propose for jumping between modes, we shall carry out an initial search for modes of the distribution π using optimization. One way of doing this is the method of *simulated annealing* (Kirkpatrick et al. (1983)). Simulated annealing is usually applied to MCMC sampling to find the global maximum of a function or distribution. Therefore, first, we are going to apply it as such, and illustrate the concept using a simple example. Later on, we shall discuss how we can use simulated annealing to find all possible modes instead of just the global maximum.

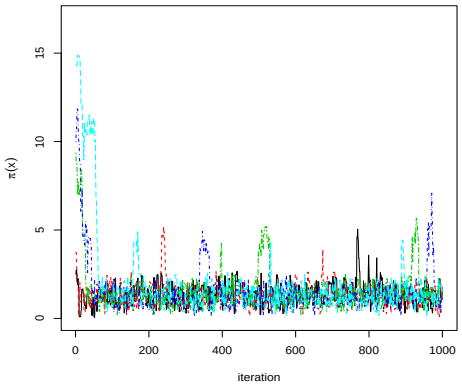
2.2.1 Description

The simulated annealing technique draws global optimization algorithms and the MCMC algorithms together. This technique originated from physics, where annealing refers to a process where a physical system is melted at a high temperature, and after that the temperature is lowered slowly to the "freezing point" so that the physical system will collapse into the lowest energy state (Kirkpatrick et al. (1983)). During the process of physical annealing, thermal equilibrium at a given temperature T is achieved when the solid has the probability of being in a state with energy E_i according to the Boltzman probability function,

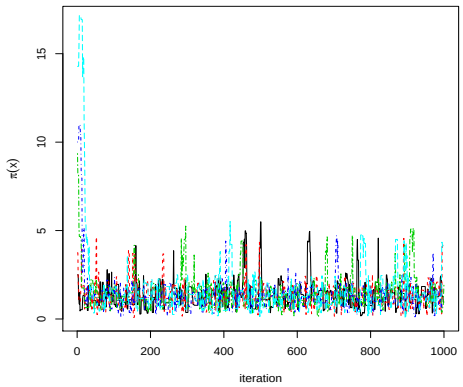
$$\mathbb{P}_T\{X = i\} = \frac{1}{Z(T)} \exp\left(\frac{-E_i}{k_B T}\right),$$



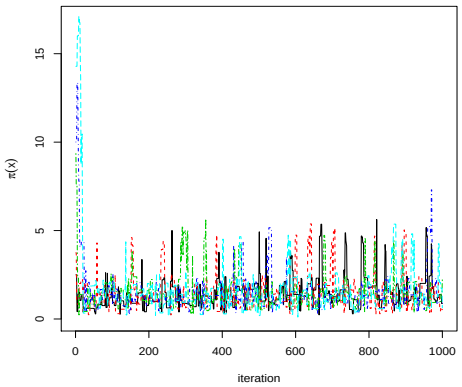
(a) $\delta = 0.1$



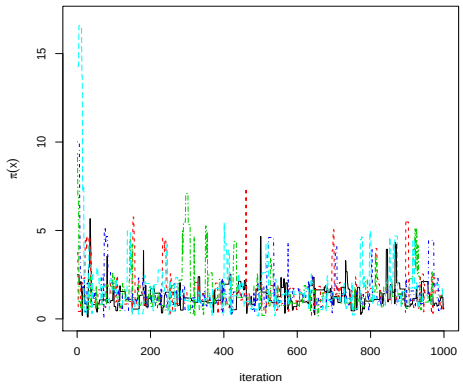
(b) $\delta = 1.0$



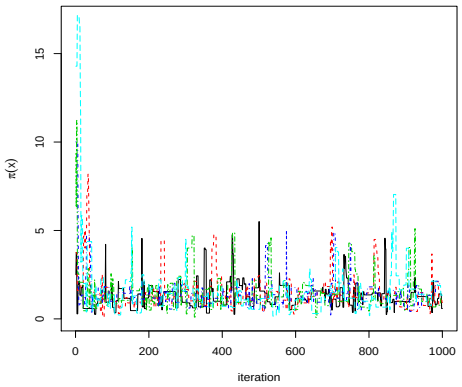
(c) $\delta = 2.0$



(d) $\delta = 3.0$



(e) $\delta = 4.0$



(f) $\delta = 5.0$

Figure 2-3: Overlapping time-series plots resulting from the MCMC sampling of $\pi(x)$, with 5 parallel sequences of length n = 1000, for the different values of δ.

where $Z(T)$ is the partition function and k_B is the Boltzman constant. As T is lowered, the Boltzman distribution collapses into the lowest energy state or states. If the possible energies correspond to the values of the objective function being minimized, then the lowest energy state will correspond to the global minimum of the objective function. For a maximization, E_i should be defined as a decreasing function of the quantity to be maximized.

2.2.2 Construction

Suppose we want to find the global maximum of our distribution of interest, $\pi(x)$. Note that finding the global maximum of $\pi(x)$ is equivalent to finding the global minimum of $-\ln(\pi(x))$. If $\pi(x)$ represents the distribution at temperature $T = 1$, the version at temperature T is

$$\pi_T(x) \propto \exp\left\{-\frac{(-\ln(\pi(x)))}{T}\right\} = \pi(x)^{1/T}.$$

First, we decide on the temperature schedule $T(t)$ as a function of iteration t. Then, at each t, the update is done as if the target distribution is proportional to $\pi(x)^{1/T(t)}$. Therefore, the MCMC sampling algorithm remains the same except the acceptance probability at iteration t is now

$$\alpha(x, y) = \min\left\{1, \frac{\pi(y)^{1/T(t)}\, q(y, x)}{\pi(x)^{1/T(t)}\, q(x, y)}\right\}.$$

At the beginning part of the sampling, where $T > 1$, the target distribution is "heated up" and flattened, so it is easier for the MCMC sampler to move around the support freely to explore. This is because the value of $(\pi(y)/\pi(x))^{1/T}$ is closer to one when T is high. Heating up the target distribution is important to ensure the algorithm will not get stuck in any particular maxima or any initial values in the beginning, especially in the case of jagged distributions, and it will depend on our choice of function $T(t)$. After this modified MCMC sampling has been run for a long time, and as the temperature goes down to (nearly) zero so that the target distribution sharpens, the intention is that the sampler should give us values that correspond to the global maximum.

The choice of the temperature function T varies with application. We shall use the logarithmic form of temperature function suggested by Hajek (1988), where $T(t) = c/\ln(t + 1)$. In his paper, Hajek (1988) showed how c could be chosen to guarantee convergence to the global minimum of $-\ln(\pi(x))$ for the case where X is discrete. Let us say that state y is reachable at height d from state x if there is a sequence of states $x_0 = x, x_1, \ldots, x_k = y$ for some $k \geq 1$ such that x_{t+1} is a neighbouring point of x_t for $0 \leq t < k$ and $-\ln(\pi(x_t)) \leq d$ for $0 \leq t \leq k$. Here a neighbouring point is a state that can be reached with probability greater than zero. Let the depth of a local minimum at state x be defined to be the smallest number d,

$d > 0$, such that some state y with $-\ln(\pi(y)) < -\ln(\pi(x))$ can be reached from x at height $-\ln(\pi(x)) + d$, and let d^* be the depth of the deepest local minimum which is not a global minimum state. Hajek (1988) proved that using the temperature function $T(t) = c/\ln(t+1)$ will make the final state of the algorithm converge in probability to the globally minimum state if c is greater than or equal to d^*. This result can also be extended to the continuous case. Using this logarithmic temperature function, we shall illustrate how well the simulated annealing performs when finding the global maximum of a distribution.

2.2.3 Application

Consider the problem of finding the global maximum of the distribution $\pi(x)$ as defined by equation (2.1.11) in Example 1, section 2.1.7. We shall see whether the simulated annealing technique using Hajek's suggestion for the temperature function will provide us with a method to move between the modes, so that the correct values for the global maximum can be found. Using $T(t) = c/\ln(t+1)$, and the proposal distribution $q(y|x) \sim \mathrm{U}(x-\delta, x+\delta)$, we simulate a chain of length $n = 100{,}000$. We start chains in two different places, and note if the chain ends in the global maximum. We repeat this for 100 different chains from each starting point for various values of c, with $\delta =$ 0.5, 1.0 and 2.0. Results are shown in Table 2.1. From the true distribution as shown in Figure 2-1, we can calculate the value of the depth d^* to be 1.8 for $\delta = 0.5$, 0.4 for $\delta = 1.0$, and 0 for $\delta = 2.0$.

If we start the chain from the top of the local maximum at $x = 4.25$, we can see from Table 2.1(a) that for values of c which are much bigger than the calculated value of d^*, there is a very high probability that the chain will end up in the global maximum, while for values of c which are much smaller than the calculated value of d^*, there is a very low probability that the chain will end up in the global maximum. However, the results are different for values of c which are close to the calculated value of d^*. The theory states that as n goes to infinity the proportion of times the chain will end up in the global maximum will jump from 0 to 100 at $c = d^*$. However, we can see from Table 2.1 that the transition is more steady. If we start the chain from the top of the global maximum, we can see from Table 2.1(b) that for the given values of c, there is a very low probability that the chain will end up in the local maximum. Overall, Hajek's (1988) theory, which predicts that using simulated annealing with temperature function $T(t) = c/\ln(t+1)$ with the condition that c be greater than or equal to d^* will result in the algorithm finding the global maximum, holds fairly well for our example. However, in practice a really effective value of c seems to be higher than the theoretical d^*.

Our aim in this example is to find the global maximum of $\pi(x)$. To this end, using

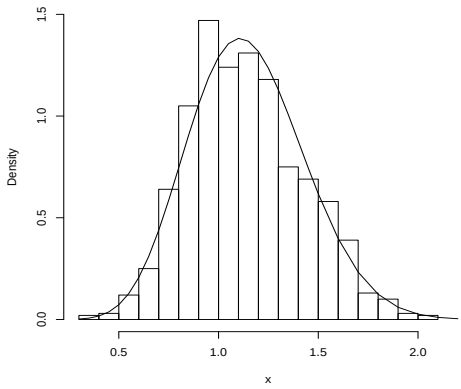
c	$\delta = 0.5$ $d^* = 1.8$	$\delta = 1.0$ $d^* = 0.4$	$\delta = 2.0$ $d^* = 0.0$
3.0	93	100	100
2.8	76	100	100
2.5	53	100	100
2.2	36	100	100
2.0	14	100	100
1.8	14	100	100
1.5	6	100	100
1.2	1	100	100
1.0	0	100	100
0.9	0	100	100
0.8	0	96	100
0.7	0	76	100
0.6	0	52	100
0.5	0	18	100
0.4	0	7	100
0.3	0	1	100
0.2	0	0	100
0.1	0	0	100

(a) Chain starts from the top of the highest local maximum

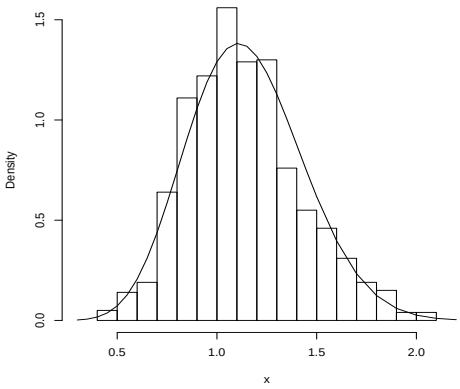
c	$\delta = 0.5$ $d^* = 1.8$	$\delta = 1.0$ $d^* = 0.4$	$\delta = 2.0$ $d^* = 0.0$
3.0	100	100	100
2.0	100	100	100
1.0	100	100	100
0.5	100	100	100
0.2	100	100	100
0.1	100	100	100

(b) Chain starts from the top of the global maximum

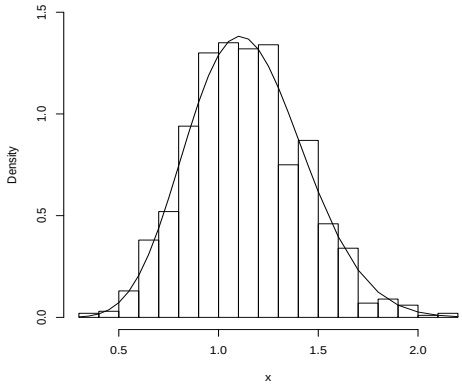
Table 2.1: Number of times (out of 100) the simulated chain of length $n = 100,000$ ends in the global maximum, where simulated annealing is carried out using different values of c and δ.



(a) $\delta = 1.0$



(b) $\delta = 2.0$



(c) $\delta = 3.0$

Figure 2-4: Histograms of the last value of each 1,000 different simulated MCMC chains of length $n = 1$ million for the different values of δ. In each case the function $\pi_T(x) = k' \, \pi(x)^{1/T(t)}$ at $t = 1$ million is superimposed.

$T(t) = c/\ln(t+1)$, with $c = 3.0$, and the proposal distribution $q(y|x) \sim \mathrm{U}(-\delta, \delta)$, we simulated a chain of length $n = 1$ million, and took the last value of the chain. We repeat this for 1,000 different chains, with $\delta = 1.0$, 2.0 and 3.0. The value $c = 3.0$ is enough to satisfy Hajek's (1998) condition for $\delta = 1.0$, 2.0 and 3.0. The histograms in Figure 2-4 show that with $c = 3.0$, the distribution of the global maximum for $\delta = 1.0$, 2.0, and 3.0 are fairly similar, and it is fairly close to the true distribution of $\pi_T(x) \propto \pi(x)^{1/T(t)}$ at $t = n = 1$ million. This shows that for any value of δ, as long as $c > d^*$ then simulated annealing works very well in finding a value in the mode at which the global maximum of the distribution occurs. However, since the last value of the chain is not usually at the very top of the mode, an additional stage of local hill-climbing is desirable.

2.2.4 Using simulated annealing to find all possible modes

In the method we shall propose for mode jumping in MCMC, we shall require an initial search for the modes, which is able to explore the distribution's whole support for all

possible modes. This is especially difficult to do if the distribution is multi-modal, or if it has a jagged shape. However, when using simulated annealing, the high temperature at the beginning heats up and flattens the target distribution, so the chain is able to move around and is able to explore the distribution's full support. In our search for all possible modes, the key is the value of c. We have to choose a suitable value of c so that the chain will sometimes end up trapped in the local maximum instead of escaping to the global maxima. Then if we use multiple starting points, chosen randomly or systematically to cover the whole state space, repeated short runs with a low value of c should give a high probability of finding all possible modes. We shall demonstrate in later examples how use of repeated short runs of fast simulated annealing can achieve this goal.

Chapter 3

Mode Jumping Methods in MCMC

3.1 Introduction

Markov chain Monte Carlo algorithms generate samples from a target distribution π by simulating a Markov chain. However, proposing small changes in the state vector in each iteration causes problems for multi-modal distributions as moves between modes are rare, and this results in poor mixing and thus slow convergence of the distribution of the Markov chain to the target distribution. This has led to different methods being proposed to obtain Markov chains with more frequent jumps between modes, and thus with better mixing and faster convergence.

Some of the proposed methods aim to improve mixing by modifying the stationary distribution π of the Markov chain and re-weighting the samples from this Markov chain to approximate π. The modified version of π is usually chosen so that it is easier for the sampling methods to move around the sample space. An example of this is a distribution which is proportional to $\pi(x)^{1/T}$, where T is chosen to be greater than one (see Jennison (1993)). A few examples of methods which aim to improve mixing by modifying the stationary distribution π are the Metropolis-coupled MCMC (Geyer (1991)), simulated tempering (Marinari & Parisi (1992); Geyer & Thompson (1995)) and tempered transitions (Neal (1996)).

In *Metropolis-coupled MCMC* (or MCMCMC), Geyer (1991) proposes running in parallel m MCMC chains, $X^{(1)}, \ldots, X^{(m)}$, with different stationary distributions $\pi_1, \ldots, \pi_m$ respectively, where $\pi_1 = \pi$ and $\{\pi_i; i > 1\}$ are chosen so that pairs π_i and π_{i+1} are close to each other. For example, we can use $\pi_i(x) \propto \pi(x)^{1/T_i}$, where T_i is chosen to increase steadily with i. One iteration comprise an update of each chain according to its associated distribution. After each iteration, the method

attempts to swap the states of two neighbouring chains using a Metropolis-Hastings step. Here, the stationary distribution of the set of m chains taken together is given by $\pi(x^{(1)}, \ldots, x^{(m)}) \propto \pi_1(x^{(1)}) \times \ldots \times \pi_m(x^{(m)})$. Only the output from chain 1 is kept, while the rest are discarded.

Simulated tempering (Marinari & Parisi (1992); Geyer & Thompson (1995)) is closely related to the Metropolis-coupled MCMC approach. But in this case, the m MCMC samplers with different stationary distributions are run in series and randomly interchanged. This method will then produce one long chain which is embedded with variable length runs from each sampler and occasional switches between samplers. Let π_i denote the stationary distribution of the i^{th} sampler, where $\pi_1 = \pi$, and let I_t indicate the value of the current sampler i at iteration t of the chain; the state of the chain at time t consists of the pair (X_t, I_t). In each iteration t of the simulated tempering, X_t is updated using sampler I_t, and then I_t is updated using a Metropolis-Hastings step. The stationary distribution of the simulated tempering chain is $\pi(x, i) \propto c_i \pi_i(x)$, where c_i are constants which may need to be chosen carefully to achieve a suitable division of probability across the set of values for i. At the end of the run only the samples $\{X_t\}$ where $I_t = 1$ are kept. One important thing to note is that this method needs at least a rough estimation of normalizing constants, which can be tricky to do. Geyer & Thompson (1995) suggests that a preliminary run of the Metropolis-coupled MCMC can be used to estimate the normalizing constants.

Tempered transitions (Neal (1996)) is similar to simulated tempering. However, instead of randomly interchanging the various MCMC samplers, the interchange will be fixed and systematic. Let the series of distributions be π_i, $i = 0, \ldots, m$, where $\pi_0 = \pi$. We define m pairs of base transition kernels, $\hat{T}_i$ and $\check{T}_i$, such that

$$\pi_i(x)\hat{T}_i(x, x') = \check{T}_i(x', x)\pi_i(x'),$$

for $i = 1, \ldots, m$. In each iteration, if the current value is $\hat{x}_0$ the proposal $\check{x}_0$ is found by applying the base transitions in the sequence $\hat{T}_1 \ldots \hat{T}_m \check{T}_m \ldots \check{T}_1$ to get $\hat{x}_1$, …, $\hat{x}_{m-1}$, $\bar{x}_m$, $\check{x}_{m-1}$, …, $\check{x}_0$. Then the proposal $\check{x}_0$ will be accepted using the Metropolis-Hastings step where the acceptance probability is

$$\alpha(\hat{x}_0, \check{x}_0) = \min\left\{1, \frac{\pi_1(\hat{x}_0)}{\pi_0(\hat{x}_0)} \ldots \frac{\pi_m(\hat{x}_{m-1})}{\pi_{m-1}(\hat{x}_{m-1})} \frac{\pi_{m-1}(\check{x}_{m-1})}{\pi_m(\check{x}_{m-1})} \ldots \frac{\pi_0(\check{x}_0)}{\pi_1(\check{x}_0)}\right\}.$$

One advantage that tempered transitions has over simulated tempering is that this method does not require estimation of the normalizing constants. However, the disadvantage of this method is the cost of sampling $2m$ times in each iteration. Neal (1996) showed that in simple problems tempered transitions is as efficient as simulated tempering; in complex problems, the performance of tempered transitions

and simulated tempering would depend on the choice of base transition kernels and samplers π_i respectively. Therefore Neal (1996) concludes that "it is not clear which method will perform best on typical problems".

Note that in the methods mentioned above, which we shall call tempering methods, the target distribution is 'heated up' and flattened so it is easier for the chain to move around the support and consequently jump between the modes; these methods can also then be related to simulated annealing. However, in cases where the target distribution has modes that are very sharp and far from each other, these tempered methods may need very hot temperatures and many steps, i.e., a large value of m, to move between the modes. This is because the success of these methods will depend on what happens in the area between the modes, where values of $\pi(x)$ are extremely small.

Alternatively, there are other kind of methods which try to improve mixing by introducing special moves which try to jump between modes. The research reported in this thesis concerns further development and application of this kind of methods. The main example of this kind was proposed by Tjelmeland & Hegstad (2001). In each iteration of their method, a big jump is followed by hill-climbing to the nearest local maximum, and a local distribution is fitted there which becomes the proposal distribution for that step. A reverse move which contains a big jump in the opposite direction is constructed in determining the acceptance probability of each proposal. In particular, this method specifies how optimization for local maxima of the target distribution can be incorporated in the specification of the Markov chain to obtain a chain with frequent jumps between modes.

Our aim is to propose a mode jumping approach which has some features of the Tjelmeland & Hegstad (2001) method but is better in other ways. Our mode jumping approach will use an *initial exploration* of the target distribution's support to find information about the local maxima, i.e., modes, and then use this information to jump between the modes effectively. This idea resembles the idea in *adaptive MCMC*, where one makes use of previously sampled states in defining an adaptive proposal distribution, i.e., one dynamically alters the proposal distribution based on information from the chain's history (see Andrieu & Moulines (2006), Atchade & Rosenthal (2005), Gasemyr (2003), Gilks et al. (1998), Haario et al. (2001), Haario et al. (2005), Mykland et al. (1995) and Sahu & Zhigljavsky (2003)). However, instead of learning about the target distribution and sampling at the same time like in adaptive MCMC, our initial exploration is focused on only learning about the target distribution by finding the modes. Before we go into details about our mode jumping approach, we first discuss Tjelmeland & Hegstad's mode jumping method.

3.2 The Tjelmeland & Hegstad (2001) mode jumping method

3.2.1 Methodology

Tjelmeland & Hegstad (2001) defined their mode jumping method for a continuous target distribution on $\mathbb{R}^n$. To sample from the target distribution $\pi(x)$, the method combines two types of update: 1) local changes; and 2) a mode jumping step. In the first type of move, local changes are proposed via the usual Metropolis-Hastings steps using a proposal distribution with comparatively small variance. In the second type of move, a mode jump is executed by taking a large step from the current state x, moving deterministically to a local maximum of π, approximating the distribution π locally at this mode, proposing a new state y from this distribution, and accepting this with a suitably defined probability $\alpha(x, y)$.

The "large step" of the mode jumping update can be described as the addition of a displacement ϕ to the current state, where ϕ is generated from a density $f(\phi)$ on $\mathbb{R}^n$. This density must be symmetric, i.e., $f(-\phi) = f(\phi)$ for all ϕ. It is convenient to think of the pair of values ϕ and $-\phi$, as defining a "move type", as described in section 2.1.4, so that we can proceed by ensuring detailed balance for moves of a particular "type". Let $r(x) = -\ln(\pi(x))$, and let $\mu(z)$ be the value of the minimum found in a deterministic local minimization of $r(x)$ starting at z. We define the Hessian matrix for r at x to be the $n \times n$ matrix

$$\nabla^2 r(x) = \left\{ \frac{\partial^2}{\partial x^{(i)}\, \partial x^{(j)}}\, r(x) \right\},$$

and we denote the inverse of this Hessian matrix evaluated at $\mu(z)$ by $\Sigma(z)$. We start by generating ϕ randomly from $f(\phi)$. Then the mode jumping algorithm for a given pair $(-\phi, \phi)$ consists of four steps:

1. Go from the current state x to $T_0(x, \phi) = x + \phi$.

2. Locate a high probability area by a deterministic local minimization of $r(x)$ starting at $T_0(x, \phi)$, producing the result $\mu(T_0(x, \phi)$. Sample y from

$$\mathrm{N}_n(\mu(T_0(x, \phi)), \Sigma(T_0(x, \phi))),$$

   where $N_n(\mu, \Sigma)$ denotes a n-variate normal distribution with mean μ and covariance Σ. Let $q_0^{\phi}(x, y)$ denote this density from which y is sampled.

3. Consider a reverse jump from y to $T_1(y, \phi) = y - \phi$, and perform another local minimization of $r(x)$, according to exactly the same algorithm used in the forward

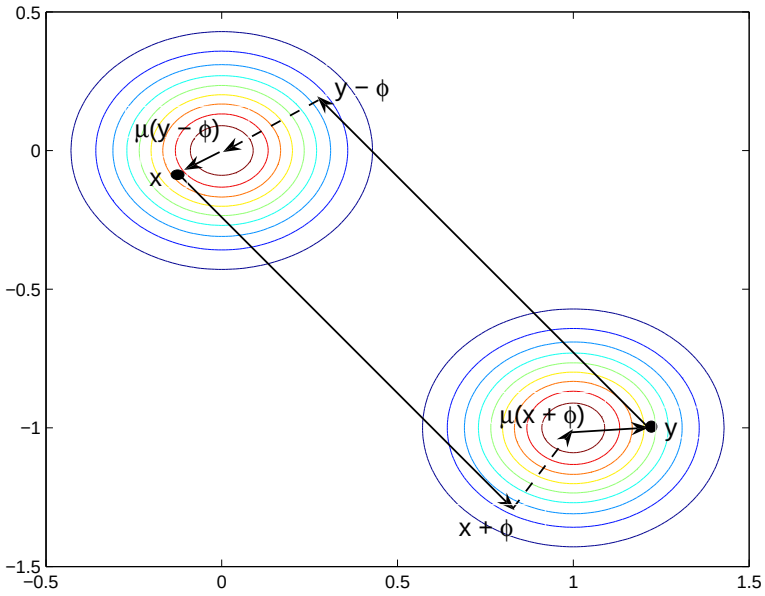


Figure 3-1: An illustration of the Tjelmeland & Hegstad (2001) mode jumping method.

step but starting at $T_1(y, \phi)$. Define $q_1^{\phi}(y, x)$ to be the density of

$$\mathrm{N}_n(\mu(T_1(y, \phi)), \Sigma(T_1(y, \phi)))$$

at x.

4. Accept the proposal to move from x to y with probability

$$\alpha(x, y) = \min\left(1, \frac{\pi(y) q_1^{\phi}(y, x)}{\pi(x) q_0^{\phi}(x, y)}\right).$$

   Otherwise, stay in state x.

An illustration of the Tjelmeland & Hegstad (2001) mode jumping proposal is given in Figure 3-1. We can prove that the Tjelmeland & Hegstad (2001) mode jumping method produces a transition kernel P_{ϕ} that satisfies detailed balance within move type ϕ by the same arguments used in section 2.1.3 to prove detailed balance for the standard Metropolis-Hastings algorithm. If the transition kernel for each move type satisfies detailed balance, then the overall transition kernel satisfies general balance (for the detailed proof, see section 2.1.4). Next, we are going show how this method works in an example.

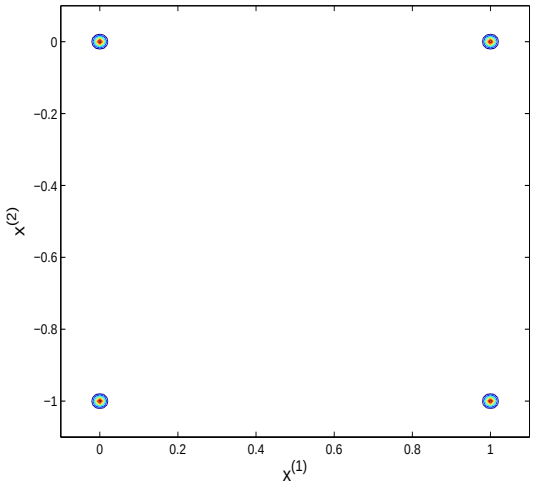


Figure 3-2: The true distribution for a mixture of 4 bivariate normal distributions on all of $\mathbb{R}^2$.

3.2.2 Example 2: Mixture of Gaussian distributions

We shall find it convenient to use the notation $\mathrm{N}_n(\boldsymbol{\mu}, \Sigma)(\boldsymbol{x})$ to denote the density at $\boldsymbol{x}$ of an n-dimensional multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix Σ. Suppose the target distribution is a mixture of four normal distributions on $\mathbb{R}^2$, and is given by

$$\pi(\boldsymbol{x}) = \sum_{i=1}^{4} \omega_i \mathrm{N}_2(\boldsymbol{\mu}_i, \Sigma_i)(\boldsymbol{x}), \tag{3.2.1}$$

where $\omega_i = 1/4$ for $i = 1, \ldots, 4$, the locations of the modes, $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_4$, are $(0, 0)$, $(1, 0)$, $(0, -1)$ and $(1, -1)$ respectively, and the covariance matrix Σ_i is

$$\Sigma_i = \begin{pmatrix} 0.01^2 & 0 \\ 0 & 0.01^2 \end{pmatrix},$$

for $i = 1, \ldots, 4$. With these choices $\pi(\boldsymbol{x})$ has 4 well separated modes, as shown in Figure 3-2.

3.2.3 Implementation of the Tjelmeland & Hegstad (2001) method and results

Suppose we want to sample from the target distribution in Example 2. When using the Tjelmeland & Hegstad (2001) mode jumping method to sample from $\pi(\boldsymbol{x})$ as defined by equation (3.2.1), we alternate between five local Metropolis-Hastings steps and one mode jumping step. The total of these six steps we call one iteration. In the Metropolis-

Hastings steps, the proposal distribution is

$$q(\boldsymbol{x}, \boldsymbol{y}) \sim \mathrm{N}_2(\boldsymbol{x}, \Sigma)(\boldsymbol{y}), \qquad \text{with} \quad \Sigma = \begin{pmatrix} 0.001^2 & 0 \\ 0 & 0.001^2 \end{pmatrix}.$$

In the mode jumping algorithm, we generate an initial displacement

$$\boldsymbol{\phi} \sim \mathrm{N}_2(\mathbf{0}, \Sigma)(\boldsymbol{x}), \qquad \text{with} \quad \Sigma = \begin{pmatrix} 2^2 & 0 \\ 0 & 2^2 \end{pmatrix}.$$

This is used to obtain $T_0(\boldsymbol{x}, \boldsymbol{\phi}) = \boldsymbol{x} + \boldsymbol{\phi}$ in the forward step and $T_1(\boldsymbol{x}, \boldsymbol{\phi}) = \boldsymbol{x} - \boldsymbol{\phi}$ in the reverse step. We use a quasi-Newton algorithm for the deterministic local minimization (for more information on this method, see Avriel (1976, chapter 11)). We have implemented the quasi-Newton algorithm using the NAG Fortran routine E04JYF. We use an approximate Hessian matrix to define $\Sigma(\boldsymbol{x})$ as we find the second derivative values numerically. Details of this numerical computation can be found in section 4.1.2.

The first 20 iterations of the simulated chain are shown in Figure 3-3. Note that the chain doesn't visit one of the modes. In these 20 iterations, 11 mode jumping proposals are accepted. However, out of the 11 successful jumps, one actually jumps back to the current mode. The 9 rejected proposals are actually rejected because on the reverse jump they propose a different mode from the original mode, which then produces an acceptance rate α which is very small. For example, the values for the mode jumps $T_0(\boldsymbol{x}, \boldsymbol{\phi})$ made from mode 1, which is located at $\boldsymbol{\mu}_1 = (0, 0)$, are shown in Figure 3-4. From the 11 proposals made from mode 1 using these values of $T_0(\boldsymbol{x}, \boldsymbol{\phi})$, 6 are rejected because of the problem with the reverse jump. An example of this situation is shown in Figure 3-5. Here we can see that a proposal to go to mode 3 at location $(0, -1)$ from mode 1 at location $(0, 0)$ will be rejected because the reverse move went to mode 2 at location $(1, 0)$ instead, due to the basin of attraction of the modes. The problem with the reverse jump can be expected to become worse as the number of modes increases.

When the proposals do propose the same mode as the original one on the reverse jump, the acceptance rate is very close to 100%. This is because this method is fitting a local bivariate normal (at each mode jumping step) to the true distribution which is a mixture of bivariate normal distributions. Furthermore, the quasi-Newton optimization algorithm manages to get values very close to the real modes, and the inverse of the approximate Hessian gives values very close to the real covariance. Hence, the fitted local bivariate normal is close to the true distribution at that mode, apart from the weight factor ω_i. Since the weight for the distribution at each mode, ω_i, is the same for $i = 1, \ldots, 4$, this means that in the calculation of $\alpha(\boldsymbol{x}, \boldsymbol{y})$ the value of $\pi(\boldsymbol{x})\, q_0(\boldsymbol{x}, \boldsymbol{y})$

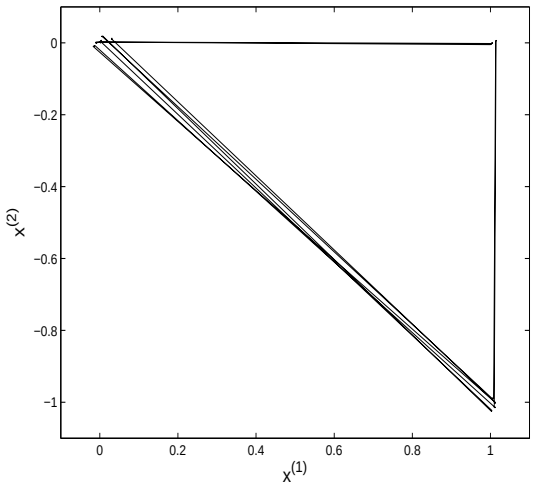


Figure 3-3: The first 20 iterations from an MCMC chain simulated using the Tjelmeland & Hegstad (2001) method.

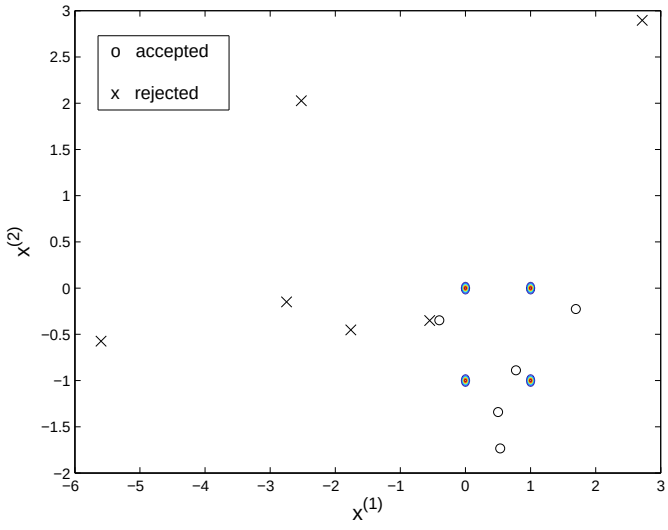


Figure 3-4: The values for the mode jumps $T_0(\boldsymbol{x}, \boldsymbol{\phi})$ made from mode 1, where mode 1 is situated at location $\boldsymbol{\mu}_1 = (0, 0)$.

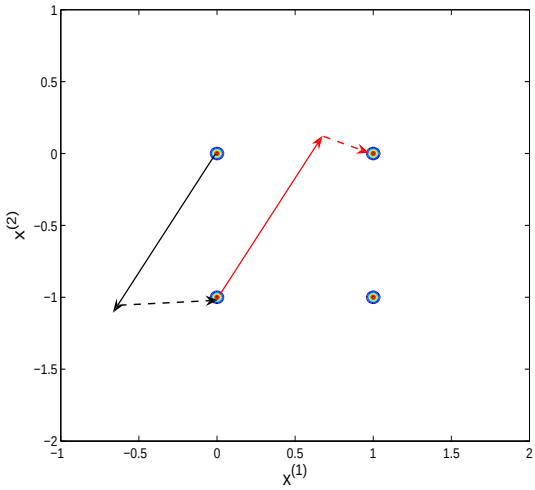


Figure 3-5: An example of the problem with the reverse jump when applying the Tjelmeland & Hegstad (2001) mode jumping method, where the black arrows represent the forward move while the red arrows represent the reverse move.

is very close to $\pi(\boldsymbol{y})\, q_1(\boldsymbol{y}, \boldsymbol{x})$, and these cancel each other out, which results in an acceptance probability which is very close to one. Note that this will not happen if ω_i is not the same for $i = 1, \ldots, 4$.

The Tjelmeland & Hegstad (2001) mode jumping method combines exploring the support of π and sampling from π at the same time. We can see that this method is not very efficient because: 1) it does not use the information about the mode locations that it has found before, but continue to search for modes and model π at the mode at each iteration; 2) its mode jumping proposal can propose values from the same mode; and 3) some of its mode jumping proposals are rejected because the reverse jump does not go back to the original mode. In their paper, Tjelmeland & Hegstad (2001) mentioned briefly that results from a prior optimization search can be combined with their method to obtain a more efficient sampler. We shall take this idea further and propose a more efficient mode jumping approach that uses an initial exploration of the target distribution to find information about the mode locations, and then uses this information to jump between the modes effectively.

3.3 A mode jumping approach based on mode locations

In this section, we shall explain in general terms a mode jumping approach which improves on the efficiency of the Tjelmeland & Hegstad (2001) mode jumping method. This approach will use an initial search to obtain prior information about the modes, and use this prior information to jump between modes. Overall, the method is divided into four stages: 1) Initial exploration; 2) Clustering; 3) Modelling; and 4) Sampling.

3.3.1 Initial exploration

The aim of the initial exploration is to *explore* the target distribution's whole support and to *find* all possible mode locations. In the initial exploration, we use optimization methods to search for the mode locations. Any kind of optimization methods can be used, but the method chosen can also depend on the target distribution. For example, for continuous target distributions one might use efficient deterministic optimization methods such as the quasi-Newton algorithm. However, optimization methods which assume that the target distribution is well behaved may behave poorly if it actually has a jagged shape. In these cases, we can use stochastic optimization methods such as simulated annealing. Note that if MCMC-type procedures are used to search for the modes, these do not have to satisfy detailed balance, as the goal is only to find the local maxima. We shall employ repeated runs using a set of starting points so that we are able to find all possible modes. Here we shall consider the use of random starting points or systematic choice covering the whole state space. If we repeat the runs l times, at the end of the initial exploration we shall have l mode locations. Since these may contain duplicates or near duplicates, the next task is to refine this set.

3.3.2 Clustering

After finding l possible mode locations in the initial exploration, we may wish to reduce this to a smaller number of distinct modes. Duplication of modes could be caused by numerical reasons — the values representing the same mode location can differ slightly due to the optimization method used in the initial exploration. We might also want to group together modes that are close to each other, as we assume that the chain can move between these modes via local steps. Reducing the number of modes is especially important in cases where mode jumping proposals are costly, so we want to make sure that the mode jumping proposals are being used to jump between modes which are difficult to move between otherwise.

We can reduce the number of modes by using *clustering*, where we group together mode locations which are similar or close to each other in distance. Here, "distance" is defined according to the problem. For example, in the case of continuous distributions the distance between two modes could be the scaled Euclidean distance between their mode locations. Many kinds of clustering algorithms are available (for more information, see Everitt et al. (2001)). After we use a clustering algorithm to cluster the l mode locations into, say m, clusters, we choose one mode location to represent each cluster. Typically, we choose the mode location with the highest probability, i.e., the highest value of $\pi(x)$ to represent a cluster. So at the end of the clustering stage, we end up with m mode locations, which we denote by η_i, $i = 1, \ldots, m$.

3.3.3 Modelling

We model each mode by fitting an approximate local distribution at each mode location. We define and fit an approximate local distribution $g_i(x)$, and possibly a weight w_i, at each mode location η_i, $i = 1, \ldots, m$. In the case of a continuous target distribution, we can fit any kind of continuous multivariate distribution that we like, depending on the problem. For example, we can use the Gaussian distribution if we suspect that the modes do not have heavy tails.

We shall also consider version of our method in which this explicit modelling is not necessary. We shall still need to generate a proposal by sampling at a selected mode and we shall present alternative methods in the sampling stage for doing this.

3.3.4 Sampling

We shall use the m mode locations in methods for *sampling* from the target distribution $\pi(x)$. To ensure that we do the sampling correctly, we are going to combine two types of update: 1) local changes; and 2) a mode jumping step. Local changes will ensure that we explore each mode, while the mode jumping step will ensure that we move between the modes. We divide methods for the mode jumping step into two types: 1) mode jumping using *modelling* (MJM); and 2) mode jumping using *differences* (MJD).

To apply our mode jumping methods, we shall define a measure of distance which allows us to compute a "nearest mode" function, $h(x) \in \{1, \ldots, m\}$, which specifies the nearest in distance of the mode locations η_i to the state x. Choosing a suitable measure of distance for $h(x)$ would depend on the problem we are applying our mode jumping method to. For example, in the case of continuous variables we can define this as the scaled Euclidean distance. Note that in the case of a tie, i.e., X has more than one nearest mode, we shall define $h(x)$ to be the jointly nearest mode with lowest index i in the list $1, \ldots, m$.

To apply our methods, we need to choose probabilities for choosing a mode to jump to. If we can estimate the weight w_i associated with the mode at location η_i, we can use these to specify

$$p_{i,j} = \frac{w_j}{\sum_{k \neq i} w_k}, \qquad j = 1, \ldots, m, \text{ and } j \neq i,$$

as the probability of proposing a jump to mode j when currently at mode i. If estimated weights are not available, we choose to jump to a different mode with equal probability, i.e., $p_{i,j} = 1/(m-1)$ for $j \neq i$.

Mode jumping using modelling (MJM)

When using the MJM method, there are two kinds of modelling that we can use: explicit modelling or implicit modelling.

1. *Explicit modelling*

In this method, we use the approximate local distributions $g_i(x)$, $i = 1, \ldots, m$, from the modelling stage to sample from the target distribution. Since the $g_i(x)$ are explicitly modelled in the modelling stage, we refer to this approach as "explicit modelling". At the start of this method, we assume that the probabilities $p_{i,\,j}$, $i,\ j = 1, \ldots, m$ and $j \neq i$, have already been defined. Then for one mode jumping step in this method, we:

1. Determine the nearest mode to the current state x, $h(x) \in \{1, \ldots, m\}$.
2. Choose a different mode $j \in \{1, \ldots, m\} \backslash h(x)$ with probability $p_{h(x),\,j}$. We get a proposal state y by sampling from $g_j(y)$.
3. Determine the nearest mode to y, $h(y) \in \{1, \ldots, m\}$.
   (a) If $h(y) \neq j$, we reject the proposal y and stay at x.
   (b) If $h(y) = j$, we accept this proposal with probability

$$\alpha(x, y) = \min\left\{1, \frac{\pi(y)\, p_{h(y),\,h(x)}\, g_{h(x)}(x)}{\pi(x)\, p_{h(x),\,h(y)}\, g_{h(y)}(y)}\right\}.$$

This mode jumping step satisfies detailed balance, as we can prove that

$$\pi(x)\, P(x, y) = \pi(y)\, P(y, x), \qquad \text{for all } x, y,$$

as shown below. Consider any pair of states x and y in state space S with $y \neq x$. It follows from the definition that a successful move require a jump between modes $h(x)$ and $h(y)$. Then,

$$\begin{aligned}
\pi(x)\, P(x, y) &= \pi(x)\, p_{h(x),\,h(y)}\, g_{h(y)}(y)\, \alpha(x, y) \\
&= \pi(x)\, p_{h(x),\,h(y)}\, g_{h(y)}(y)\, \min\left\{1, \frac{\pi(y)\, p_{h(y),\,h(x)}\, g_{h(x)}(x)}{\pi(x)\, p_{h(x),\,h(y)}\, g_{h(y)}(y)}\right\} \\
&= \pi(y)\, p_{h(y),\,h(x)}\, g_{h(x)}(x)\, \min\left\{1, \frac{\pi(x)\, p_{h(x),\,h(y)}\, g_{h(y)}(y)}{\pi(y)\, p_{h(y),\,h(x)}\, g_{h(x)}(x)}\right\} \\
&= \pi(y)\, p_{h(y),\,h(x)}\, g_{h(x)}(x)\, \alpha(y, x) \\
&= \pi(y)\, P(y, x),
\end{aligned}$$

a similar argument to that for the standard Metropolis-Hastings algorithm. For any pair of states x and y in S with $y = x$, it is true trivially that $\pi(x)\, P(x, y) = \pi(y)\, P(y, x)$.

Note that in this new approach we only model each mode once before starting sampling, i.e., in the modelling stage. In contrast, the Tjelmeland & Hegstad (2001) method has to locate the mode and model the distribution there in every mode jumping step.

2. *Implicit modelling*

In some cases, it is not feasible to model each mode explicitly, for example when the variable X has a large number of discrete components. It is, however, possible to produce a randomly sampled value in the vicinity of a mode by performing one cycle of the Gibbs sampler with the corresponding location of that mode as the starting value (Sharp (2003), chapter 5). Since this value is intended to be a likely realization under a local model for π at the mode, had we been able to fit one, we refer to this approach as "implicit modelling".

At the start of this method, we assume that we know η_i, $i = 1, \ldots, m$, from the initial exploration and clustering process, and that the probabilities $p_{i,\,j}$, $i, j = 1, \ldots, m$ and $j \neq i$, have already been defined. We shall define $q_i(x)$ to be the probability of η_i changing to the state x via one cycle of the Gibbs sampler. Then for one mode jumping step in this method, we:

1. Determine the nearest mode to the current state x, $h(x) \in \{1, \ldots, m\}$.
2. Choose a different mode $j \in \{1, \ldots, m\} \backslash h(x)$ with probability $p_{h(x),\,j}$. We perform one cycle of the Gibbs sampler, starting from η_j to get a proposal y, and compute the corresponding probability $q_j(y)$.
3. Determine the nearest mode to y, $h(y) \in \{1, \ldots, m\}$.
   (a) If $h(y) \neq j$, we reject the proposal y and stay at x.
   (b) If $h(y) = j$, we accept this proposal with probability

$$\alpha(x, y) = \min\left\{1, \frac{\pi(y)\, p_{h(y),\,h(x)}\, q_{h(x)}(x)}{\pi(x)\, p_{h(x),\,h(y)}\, q_{h(y)}(y)}\right\}.$$

This mode jumping step satisfies detailed balance following the same argument as in the explicit modelling case but replacing the function g_i with q_i.

Mode jumping using differences (MJD)

We have found an alternative form of mode jumping using the differences between mode locations to be useful in cases where the difference between two modes only involves a subset of variables. The advantage of this approach is that we then only need to update

this subset of variables when making a proposal and can leave the rest unchanged. Let $d_{i,j}$ be the difference between two mode locations η_i and η_j. For example, in the continuous case we can define $d_{i,j}$ to be $\eta_j - \eta_i$, and $d_{j,i} = -d_{i,j}$. We shall use these differences to jump between the modes. Let $a \oplus b$ denote the process of adding a value b to a value a according to some predefined rules, which will be defined according to the application. For example, in the case of a continuous multivariate target distribution we might define $a \oplus b = a + b$. In cases where X has a bounded state space (at either side or both), a modification of this definition is necessary, for example, for the state space in binary images where $S = \{0, 1\}$, we can define $a \oplus b = \max(\min(a + b, 1), 0)$ (please refer to chapter 5 for more details).

At the start of this method, we assume that the probabilities $p_{i,j}$, i, $j = 1, \ldots, m$ and $j \neq i$, have already been defined. We shall define the operator $\oplus$ and a random perturbation process, where we define $q(x, y)$ to be the probability of moving from a state x to a state y via this process. Then for one mode jumping step in this method, we:

1. Determine the nearest mode location to the current state x, $h(x) \in \{1, \ldots, m\}$.
2. Choose a different mode location $j \in \{1, \ldots, m\} \backslash h(x)$ with probability $p_{h(x),j}$.
3. Define $y' = x \oplus d_{h(x),j}$. We then perturb y' randomly using the defined random perturbation process to get the proposal y and the corresponding probability $q(y', y)$.
4. Determine the nearest mode to y, $h(y) \in \{1, \ldots, m\}$.
   (a) If $h(y) \neq j$, we reject the proposal y and stay at x.
   (b) If $h(y) = j$, we accept this proposal with probability

$$\alpha(x, y) \quad = \quad \min\left\{1, \frac{\pi(y)\, p_{h(y),h(x)}\, q(x', x)}{\pi(x)\, p_{h(x),h(y)}\, q_(y', y)}\right\}.$$

This mode jumping step satisfies detailed balance, as we can prove that

$$\pi(x)\, P(x, y) = \pi(y)\, P(y, x), \qquad \forall\, x, y,$$

as shown below. For any pair of states x and y in state space S with $y \neq x$,

$$
\begin{aligned}
\pi(x)\,P(x,y) &= \pi(x)\,p_{h(x),\,h(y)}\,q(y',y)\,\alpha(x,y) \\
&= \pi(x)\,p_{h(x),\,h(y)}\,q(y',y)\,\min\left\{1,\frac{\pi(y)\,p_{h(y),\,h(x)}\,q(x',x)}{\pi(x)\,p_{h(x),\,h(y)}\,q(y',y)}\right\} \\
&= \pi(y)\,p_{h(y),\,h(x)}\,q(x',x)\,\min\left\{1,\frac{\pi(x)\,p_{h(x),\,h(y)}\,q(y',y)}{\pi(y)\,p_{h(y),\,h(x)}\,q(x',x)}\right\} \\
&= \pi(y)\,p_{h(y),\,h(x)}\,q(x',x)\,\alpha(y,x) \\
&= \pi(y)\,P(y,x).
\end{aligned}
$$

For any pair of states x and y in S with $y = x$, it is true trivially that $\pi(x)\,P(x,y) = \pi(y)\,P(y,x)$. Since in this method we are not assuming or fitting any distribution, the modelling stage will not be needed.

The random perturbation process will be defined according to the application. For example, in the case of a continuous multivariate target distribution we might define the proposal y to be $y = y' + \epsilon$ where $\epsilon \sim \mathrm{N}_2(\mathbf{0}, \Sigma)$ with some small predefined variance matrix Σ. With these choices $q(y', y) = \mathrm{N}_2(\mathbf{0}, \Sigma)(y - y')$. We can also define the random perturbation process to be an update using one cycle of the Gibbs sampler. Note that this process can be defined to involve all components of X, or just a subset of X. We shall see more details regarding this matter in chapters 5 and 6.

An extreme version of this form of our method is where no random perturbations are added to the proposal, i.e., $y = y'$. In this case, both the probability of moving from state y' to state y and the probability of moving from state x' to state x will be equal to 1. In general, the mode jumping using differences method will propose values which are in the general vicinity of the chosen mode j, especially if π has similar shape at each mode. However, if the target distribution π has different shape and scale at each mode it will be fairly difficult for y to be accepted, and this mode jumping method might not work very well.

3.4 Considerations for efficiency and accuracy

A good sampler is efficient and produces accurate estimates. Here we define efficiency as a function of both variance of estimates and computation used per iteration. In a multi-modal distribution, an increase in the movement between the modes should mean the chain is mixing more, leading to better samples and more accurate estimates.

Using these criteria, we expect our new mode jumping methods to be better than the Tjelmeland & Hegstad (2001) method as our methods

- Find the information about the modes just once in an initial exploration;
- Propose mode jumping proposals in a more efficient way;
- Produce proposals which have higher chances of being accepted; and
- Always propose a jump to a different mode.

Our methods also have an advantage over the tempering methods discussed in section 3.1 as in the sampling stage our methods jump across modes, and do not rely on the behaviour of the target distribution in between modes. The tempering methods also take a lot of effort to move between modes, while in the sampling stage our methods propose jumping directly from one mode to another without visiting intermediate states at all.

However, one point of concern, peculiar to our mode jumping approach, is the danger of failing to find a mode in the initial exploration stage. This is especially important since our mode jumping approach only tries to find the modes in the initial exploration stage, while the other methods continue to try to find the modes all the way through. Indeed, Tjelmeland & Hegstad (2001) suggested that results from a prior optimization search have to be combined with their (original) mode jumping method because one can never be sure of having located all the important modes of $\pi(x)$ in the prior optimization search.

Nevertheless, we believe that the danger of failing to find a mode in the initial exploration stage of our method is low. We shall show this by concentrating on the continuous case and showing that asymptotic theory similar to that stated in Theorem 2.2 in section 2.1.2 can be obtained for our mode jumping method. In doing this, we consider a formulation in which resources are divided between initial exploration and sampling so that asymptotically, the length of both stages goes to infinity. The same approach can also be applied to the discrete case and the corresponding Theorem 2.1 in section 2.1.1.

We first define our initial exploration stage as a number of "runs", ending at a mode each time. Suppose in our mode jumping approach for a given value of n, we carry out $\gamma_1 n$ "runs" in the initial exploration stage, and $\gamma_2 n$ iterations of MCMC sampling in the second stage. At the end of this we obtain X, and we define its distribution to be $R^n(x|x_0)$ for a given starting state $X_0 = x_0$ of the sampling stage. We would like to show

$$||R^n(\cdot\,|x_0) - \pi(\cdot)|| \to 0 \qquad \text{as} \qquad n \to \infty,$$

for π-almost all x_0. We shall assume that the target distribution π has a finite number of modes and that the numerical error in finding the modes are negligible. We shall

also assume that in the initial exploration stage, the probability of visiting mode j some time in the future when currently at mode i is greater than zero for all i and j. We denote the outcome of the initial exploration stage by M_n, where

$$M_n = \begin{cases} 1, & \text{if we find all the modes} \\ 0, & \text{otherwise.} \end{cases}$$

Let $\mathbb{P}(M_n = 0) = \xi_n$, then since $\gamma_1 n \to \infty$ as $n \to \infty$, it follows directly from the Markov nature of the initial exploration process that

$$\xi_n \to 0 \qquad \text{as} \qquad n \to \infty.$$

Let $R_1^n(\cdot\,|x_0)$ be the conditional distribution of the final X conditioned on $M_n = 1$ and starting state $X_0 = x_0$, and $R_0^n(\cdot\,|x_0)$ be the conditional distribution of the final X conditioned on $M_n = 0$ and $X_0 = x_0$. Therefore,

$$R^n(\cdot\,|x_0) = (1 - \xi_n)\, R_1^n(\cdot\,|x_0) + \xi_n R_0^n(\cdot\,|x_0).$$

Thus, what we need to prove is:

$$||(1 - \xi_n)\, R_1^n(\cdot\,|x_0) + \xi_n R_0^n(\cdot\,|x_0) - \pi(\cdot)|| \to 0 \quad \text{as} \quad n \to \infty, \tag{3.4.1}$$

for π-almost all x_0. We shall prove this now using the definition of total variation distance given in section 2.1.2 and the fact that $|A + B + C| \leq |A| + |B| + |C|$.

$$\begin{aligned}
&||(1 - \xi_n)\, R_1^n(\cdot\,|x_0) + \xi_n\, R_0^n(\cdot\,|x_0) - \pi|| \\
&= \sup_{A \subset S} |(1 - \xi_n)\, R_1^n(A|x_0) + \xi_n R_0^n(A|x_0) - \pi(A)| \\
&= \sup_{A \subset S} |R_1^n(A|x_0) - \pi(A) - \xi_n R_1^n(A|x_0) + \xi_n R_0^n(A|x_0)| \\
&\leq \sup_{A \subset S} |R_1^n(A|x_0) - \pi(A)| + |\xi_n R_1^n(A|x_0)| + |\xi_n R_0^n(A|x_0)| \\
&\leq \sup_{A \subset S} |R_1^n(A|x_0) - \pi(A)| + \xi_n + \xi_n \\
&= \sup_{A \subset S} |R_1^n(A|x_0) - \pi(A)| + 2\xi_n \\
&= ||R_1^n(\cdot\,|x_0) - \pi(\cdot)|| + 2\xi_n.
\end{aligned}$$

Since

$$||R_1^n(\cdot\,|x_0) - \pi(\cdot)|| \to 0 \quad \text{as} \quad n \to \infty$$

for π-almost all x_0 when we apply Theorem 2.2 and $\xi_n \to 0$ as $n \to \infty$, therefore (3.4.1) is true. This establishes the desired result for the convergence of $R^n(\cdot\,|x_0)$.

As an equivalent result to the second part of Theorem 2.2, we would like to show that for any real-valued function f,

$$\mathbb{P}\left(\frac{1}{\gamma_2 n}\sum_{i=1}^{\gamma_2 n} f(X_i) \;\rightarrow\; \mathbb{E}_\pi\left(f(X)|X_0 = x_0\right)\right) = 1 \tag{3.4.2}$$

for π-almost all x_0. In order to obtain this result, we define both the initial exploration and sampling stages in a specific "nested" way as $n \rightarrow \infty$. Again, we assume that the target distribution π has a finite number of modes and that the numerical error in finding the modes are negligible. For the initial exploration process, we assume the sequence of runs simply grows longer as n, and therefore $\gamma_1 n$, increases. Therefore, this process will generate a growing set of mode values. At some finite point $n = n^*$ in the initial exploration process we shall find all the modes, and for any $n > n^*$ we shall have all the modes and these modes remain the same.

We define the sampling process as being "nested" in the following ways: firstly, the same starting state $X_0 = x_0$ is used for all n and secondly, the updates are implemented using the same sequence of random numbers. The chains produced by this process are, therefore, "coupled". The samples produced will still depend on the set of modes that we found in the initial exploration stage. However, if the same set of modes is found during the initial exploration stage for $n = n_1$ and $n = n_2 > n_1$, then the sampling run for $n = n_1$ will also form the initial part of the longer sampling run for $n = n_2$. Now consider values of $n > n^*$, so in all these cases the full set of modes is found in the initial exploration stage. The sampling process produces a long chain of length $\gamma_2 n$ and these chains are nested as n increases. Standard theory of the original Theorem 2.2 can be applied to this situation as $n \rightarrow \infty$ to prove (3.4.2).

In practice, we feel that in small, non-infinite samples, a small amount of our initial exploration is as good as Tjelmeland & Hegstad's (2001) continued exploration. This is because the Tjelmeland & Hegstad (2001) method uses random movements to search for a mode and sample at the same time (and has to maintain detailed balance), while the initial exploration stage has a more direct objective: search for the modes. We are not restricted by the methods that we can use to search for the modes (we do not have to maintain detailed balance and so on) and we can concentrate all our efforts on searching for the modes in the most effective way. Our search for the modes is also more systematic. our initial exploration We shall see a more detailed comparison in section 4.2.4 when we apply both methods to a specific example.

The matters that we have discussed above show that there is a need for a suitable balance between initial exploration and sampling. Therefore, we cannot overlook the importance of *many* short optimization runs in the initial exploration stage. But with

this in place, it will not be necessary to follow Tjelmeland & Hegstad's (2001) plan of incorporating additional exploration in the sampling stage.

Note that the performance of our mode jumping approach does depend on making good choices in each of the four stages. For example, we have to make good choices when selecting the starting points and optimization method for the initial exploration stage; we can choose these based on the information that we have about the problem. We shall see this further when we apply our methods to specific examples in the coming chapters.

Chapter 4

Application I: Continuous Variables

4.1 General methodology for continuous variables

In this chapter, we shall discuss how we can apply our mode jumping approach to sample from a target distribution with continuous variables in $\mathbb{R}^p$ and then compare, in an example, the performance of this method with that of Tjelmeland & Hegstad (2001). In the course of this example, we shall address directly the issues connected to our mode jumping approach that were discussed in section 3.4. To apply our mode jumping approach to sample from a distribution with continuous variables, we shall go through the four stages:

1. Initial exploration, where we try to find all possible locations of the modes.
2. Clustering, where we cluster the mode locations and find a single mode location to represent each cluster.
3. Modelling, where we fit an approximate model at each of the mode locations obtained from the clustering stage.
4. Sampling, where we use the models from the modelling stage to jump between the modes.

While the initial exploration and sampling stages remain the same as described in section 3.3, the clustering and modelling stages need to be defined more specifically for the general case of continuous variables. Note that we denote the target distribution by $\pi(\boldsymbol{x})$ and that we only know $\pi(\boldsymbol{x})$ up to a normalizing constant, i.e., we know $\pi(\boldsymbol{x}) = c\,\psi(\boldsymbol{x})$ where $\psi(\boldsymbol{x})$ is known but c is unknown.

4.1.1 Clustering

After finding l mode locations in the initial exploration stage, we shall use clustering to reduce this to a smaller number of distinct modes. In the continuous variables case, the main reason for doing this is to remove duplicates of the same mode. Duplication of modes could happen due to numerical reasons — the values representing the same mode location can differ slightly due to the optimization method used in the initial exploration. Then in this stage, we shall use a clustering algorithm to group the l mode locations into clusters. We need to define the "distance" measure that we are going to use when we apply our method to continuous variables. Let $\boldsymbol{X}_i$, $i = 1, \ldots, l$, denote the mode locations found, and let the observed standard deviation among these for the k^{th} component be defined as

$$s^{(k)} = \left[\sum_{i=1}^{l} \frac{(X_i^{(k)} - \bar{X}^{(k)})^2}{l-1} \right]^{1/2},$$

where $\bar{X}^{(k)} = \sum_{i=1}^{l} X_i^{(k)}/l$. For continuous variables, we define the distance between two mode locations $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$, a_{ij}, to be their scaled Euclidean distance, i.e.,

$$a_{ij} = \left\{ \sum_{k=1}^{n} \left(\frac{X_i^{(k)}}{s^{(k)}} - \frac{X_j^{(k)}}{s^{(k)}} \right)^2 \right\}^{1/2}. \tag{4.1.1}$$

Note that we are using *scaled* Euclidean distance because each component might use different units or scale of values and, without this scaling, some terms could completely dominate the sum in (4.1.1). Using this definition of distance, we compute the distance between all the l mode locations, i.e., a_{ij} for $i, j = 1, \ldots, l$.

To group the l mode locations, we use an agglomerative hierarchical clustering algorithm called the nearest-neighbour technique, or single linkage (see chapter 4 of Everitt et al. (2001)). In this algorithm, we define the distance between two clusters, i.e., two groups of points, to be the smallest distance between any two points, one from each cluster. We also define a stopping criterion, ξ, such that at the end of the algorithm the clusters will be at least ξ apart from each other. This algorithm proceeds by a series of successive fusion of the l mode locations into groups, i.e., we start with l clusters, each with a single mode location, and then at each of the following stages we merge two clusters which are closest in distance to form a larger cluster until all the clusters are at least a distance ξ apart from each other at which point the algorithm stops. We implement this algorithm using the output from a NAG Fortran routine G03ECF combined with the stopping criterion ξ imposed on it. The number of modes, m, present in the distribution is the number of clusters obtained at the end of the clustering algorithm. We shall choose the mode location with the highest value of $\pi(\boldsymbol{x})$

within a cluster to represent that cluster. At the end of the clustering stage, we shall end up with m mode locations, $\boldsymbol{\eta}_i$, $i = 1, \ldots, m$.

Note that running this clustering algorithm is not computationally intensive at all, and the amount of computation used by this algorithm is negligible when compared to the amount needed to carry out the initial exploration and sampling stages. However, the amount of computation used by this algorithm does increase quickly, in proportion to l^2.

4.1.2 Modelling

Suppose that the form of π is close to a multivariate normal distribution at each mode so, approximately,

$$\pi(\boldsymbol{x}) = \sum_{i=1}^{m} w_i \mathrm{N}_n(\boldsymbol{\eta}_i, \Sigma_i)(\boldsymbol{x})$$

where $\sum_{i=1}^{m} w_i = 1$, and therefore

$$\pi(\boldsymbol{x}) \approx w_i \mathrm{N}_n(\boldsymbol{\eta}_i, \Sigma_i)(\boldsymbol{x}), \tag{4.1.2}$$

for $\boldsymbol{x}$ near $\boldsymbol{\eta}_i$. We assume we know $\boldsymbol{\eta}_i$ from the initial search and clustering process. We therefore need to find Σ_i and w_i. Let $r(\boldsymbol{x}) = -\ln(\pi(\boldsymbol{x}))$. We can show that if expression (4.1.2) is true, then

$$\frac{\partial^2}{\partial {x^{(j)}}^2} r(\boldsymbol{x}) = (\Sigma_i^{(jj)})^{-1} \qquad \text{for } j = 1, \ldots, p$$

and

$$\frac{\partial^2}{\partial x^{(j)} \, \partial x^{(k)}} r(\boldsymbol{x}) = (\Sigma_i^{(jk)})^{-1} \qquad \text{for } j,\ k = 1, \ldots, p,\ k \neq j.$$

If it is not possible to obtain the derivatives of $r(\boldsymbol{x})$ analytically, they can be found numerically. Let $\boldsymbol{v}_j$ be a vector of length n with a value v in element j and 0 everywhere else, and let $\boldsymbol{v}_{jk}$ be a vector of length n with v in element j and k and 0 everywhere else. Also let $r'(\boldsymbol{x}) = -\ln(\psi(\boldsymbol{x}))$. Then, since $r(\boldsymbol{x}) = -\ln(c) + r'(\boldsymbol{x})$, we can approximate the second derivatives by using

$$\begin{aligned} \frac{\partial^2}{\partial {x^{(j)}}^2} r(\boldsymbol{x}) &\approx \frac{r(\boldsymbol{x} + \boldsymbol{v}_j) - 2r(\boldsymbol{x}) + r(\boldsymbol{x} - \boldsymbol{v}_j)}{v^2} \\ &= \frac{r'(\boldsymbol{x} + \boldsymbol{v}_j) - 2r'(\boldsymbol{x}) + r'(\boldsymbol{x} - \boldsymbol{v}_j)}{v^2}, \qquad \text{for } j = 1, \ldots, p, \end{aligned} \tag{4.1.3}$$

and

$$\frac{\partial^2}{\partial x^{(j)}\,\partial x^{(k)}}\, r(\boldsymbol{x}) \approx \frac{r(\boldsymbol{x}+\boldsymbol{v}_{jk}) - r(\boldsymbol{x}+\boldsymbol{v}_j) - r(\boldsymbol{x}+\boldsymbol{v}_k) + r(\boldsymbol{x})}{v^2}$$
$$= \frac{r'(\boldsymbol{x}+\boldsymbol{v}_{jk}) - r'(\boldsymbol{x}+\boldsymbol{v}_j) - r'(\boldsymbol{x}+\boldsymbol{v}_k) + r'(\boldsymbol{x})}{v^2},$$
$$\text{for } j,\ k = 1,\ldots,p, \quad k \neq j. \quad (4.1.4)$$

To find the approximate second derivative values at $\boldsymbol{\eta}_i$, we apply expressions (4.1.3) and (4.1.4) with $\boldsymbol{x} = \boldsymbol{\eta}_i$, the mean of the local approximation at mode i. However, this choice of value for $\boldsymbol{x}$ is not crucial: if $\pi(\boldsymbol{x})$ is locally multivariate normal, the second derivatives of $-\ln(\pi(\boldsymbol{x}))$ at any value of $\boldsymbol{x}$ near $\boldsymbol{\eta}_i$ will provide the elements of the inverse of Σ_i.

Let $g_i(\boldsymbol{x})$ be the density function of a normal distribution $\mathrm{N}_n(\boldsymbol{\eta}_i, \Sigma_i)$ at $\boldsymbol{x}$. If (4.1.2) is true, then

$$c\,\psi(\boldsymbol{x}) \approx w_i \mathrm{N}_n(\boldsymbol{\eta}_i, \Sigma_i)(\boldsymbol{x}),$$

for $\boldsymbol{x}$ near $\boldsymbol{\eta}_i$. Therefore

$$c\,\psi(\boldsymbol{\eta}_i) \approx w_i \mathrm{N}_n(\boldsymbol{\eta}_i, \Sigma_i)(\boldsymbol{\eta}_i)$$

and thus, the weight w_i for the mode at $\boldsymbol{\eta}_i$ is given by

$$w_i = c\,\frac{\psi(\boldsymbol{\eta}_i)}{g_i(\boldsymbol{\eta}_i)},$$

for each $i = 1,\ldots,m$. Let $q_i = \psi(\boldsymbol{\eta}_i)/g_i(\boldsymbol{\eta}_i)$. Since $\sum_{i=1}^m w_i = 1$, we can calculate w_i by using

$$w_i = \frac{q_i}{\sum_{j=1}^m q_j},$$

even though c was unknown.

4.2 Applying the MJM method to Example 2

We shall apply our MJM method to sample from the mixture of four normal distributions defined by equation (3.2.1) in section 3.2.2 and given in Figure 4-1, and compare the performance of our new method with that of the Tjelmeland & Hegstad (2001) method.

4.2.1 Implementation of the mode jumping approach

1. Initial exploration

   We do an initial exploration of the distribution's whole support to find local modes. We start off with a random value $\boldsymbol{x}_0$ where $0 < x_0^{(1)} < 1$ and

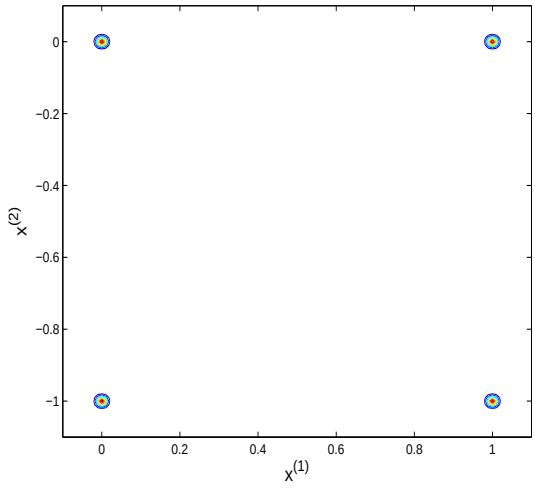


Figure 4-1: The true distribution for a mixture of 4 bivariate normal distributions on all of $\mathbb{R}^2$.

$-1 < x_0^{(2)} < 0$, and apply the quasi-Newton algorithm to the function $-\ln(\pi(\boldsymbol{x}))$ to find the nearest local maximum $\boldsymbol{x}'$. Then we make a random move to $\boldsymbol{y}$ from $\boldsymbol{x}'$ where

$$\boldsymbol{y} \sim \mathrm{N}_2(\boldsymbol{x}', \Sigma)(\boldsymbol{y}), \qquad \text{with} \quad \Sigma = \begin{pmatrix} 2^2 & 0 \\ 0 & 2^2 \end{pmatrix},$$

and use the quasi-Newton algorithm again to find the local minimum of $-\ln(\pi(\boldsymbol{x}))$. We repeat the procedure of making a random move and optimizing 100 times, and the locations of all the local maxima found form a list of possible modes. At the end of this stage we have 100 mode locations. Note that the choice of Σ would depend on the information that we have about the problem. Σ should produce steps away from $\boldsymbol{x}'$ that is big enough to move between modes.

2. Clustering

   We cluster the values from the initial exploration stage using the hierarchical clustering algorithm, as described in section 4.1.1, using scaled Euclidean distance and with $\xi = 0.01$. At the end of this stage, we have m mode locations, $\boldsymbol{\eta}_i$, $i = 1, \ldots, m$.

3. Modelling

   We fit a normal distribution at each of the mode locations found in the clustering stage and calculate the approximate weight for each accordingly, as described in section 4.1.2. Here we choose $v = 1 \times 10^{-5}$.

4. Sampling

   In the sampling stage, we alternate between two parts: 1) local changes via 5 steps of the usual Metropolis-Hastings sampling method; and 2) a mode jumping

step. The total of these 6 steps we call one iteration. In the first part, local changes are proposed via the usual Metropolis-Hastings algorithm by using a proposal distribution with comparatively small variance values, i.e., by using

$$q(\boldsymbol{x}, \boldsymbol{y}) \sim \mathrm{N}_2(\boldsymbol{x}, \Sigma)(\boldsymbol{y}), \qquad \text{with} \quad \Sigma = \begin{pmatrix} \varsigma^2 & 0 \\ 0 & \varsigma^2 \end{pmatrix}.$$

We can choose ς^2 by referring to the local model fitted in the modelling stage. Here we choose $\varsigma^2 = 0.001^2$. Note that the choice of local moves will effect the results to some extend.

In the second part, we define our "nearest mode" function $h(\boldsymbol{x}) \in \{1, \ldots, m\}$ so that it will choose the nearest mode i which has the minimum scaled Euclidean distance between $\boldsymbol{X}$ and $\boldsymbol{\eta}_i$, where we shall use the scaling values $\boldsymbol{s} = (s^{(1)}, s^{(2)})$ as defined in section 4.1.1 but only using the m mode locations found from the clustering stage. The mode jumping step then proceeds as given in section 3.3.4.

Note that to make our method comparable to the Tjelmeland & Hegstad (2001) mode jumping method, we have chosen the same number of local changes and mode jumping step per iteration and the same local proposal distribution for both methods. Implementation of the Tjelmeland & Hegstad (2001) mode jumping method follows the description given in section 3.2.3.

4.2.2 Criteria for comparing MCMC sampling methods

When using any MCMC sampling methods we want to learn more about the target distribution $\pi(\boldsymbol{x})$ by sampling from it. For example, we can use the simulated samples to estimate the mean and variance under π of a function $f(\boldsymbol{x})$, or find the probability of a certain event that we are interested in. In multi-modal distributions, the modes dominate $\pi(\boldsymbol{x})$. This means that when sampling using any MCMC sampling methods, the proportion of time the chain spends in each mode is important as it will affect the estimates of the probability that π assigns to each mode. Furthermore, mode jumping is usually the hard part of sampling from a multi-modal distribution as it is more challenging and likely to be a slower process than moving around within a mode. Therefore when we measure the performance of any MCMC sampling methods, we shall focus more on the movement of the simulated chain between the modes.

We can check the movement of the chain between the modes by using a nearest mode function $m(\boldsymbol{x})$, which finds the nearest mode to the state $\boldsymbol{x}$. In our examples, we shall define $m(\boldsymbol{x})$ so that it will choose the nearest mode i which has the minimum value of the scaled Euclidean distance between $\boldsymbol{X}$ and the real mode location $\boldsymbol{\mu}_i$, where in this case we shall use scales which are comparable to the units of each component

$X^{(i)}$, $i = 1, \ldots, p$. The nearest mode function $m(\boldsymbol{x})$ may not be quite the same as the nearest mode function $h(\boldsymbol{x})$ that was defined in previous section as $m(\boldsymbol{x})$ identifies the nearest mode using the *real* mode locations $\boldsymbol{\mu}_i$ while $h(\boldsymbol{x})$ uses the mode locations $\boldsymbol{\eta}_i$ obtained from the initial exploration and clustering stages. However, if $\boldsymbol{\eta}_i$ are very close to $\boldsymbol{\mu}_i$ then $h(\boldsymbol{x})$ is very close to $m(\boldsymbol{x})$.

Note that although the process $\{\boldsymbol{X}_t\}$ itself is Markov, the process $\{m(\boldsymbol{X}_t)\}$ may not be. We shall define a matrix $P = \{p_{ij}\}$ related to the process $\{m(\boldsymbol{X}_t)\}$ when $\boldsymbol{X}_t$ follows the stationary distribution π, where

$$\begin{aligned} p_{ij} &= \mathbb{P}(m(\boldsymbol{X}_{t+1}) = j | m(\boldsymbol{X}_t) = i) \\ &= \frac{\int_{x_t : m(x_t)=i} \mathbb{P}(m(\boldsymbol{X}_{t+1}) = j | \boldsymbol{X}_t = \boldsymbol{x}_t)\, \pi(x_t)\, \mathrm{d}x_t}{\int_{x_t : m(x_t)=i} \pi(x_t)\, \mathrm{d}x_t}. \end{aligned}$$

If we assume the process $\{m(\boldsymbol{X}_t)\}$ to be approximately Markov, then P would be the corresponding transition matrix under this approximation. The true value of P is usually unknown, but we can estimate it by running the MCMC sampler for n number of iterations, where n is large, to obtain the sample values $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ and determine the corresponding values $m(\boldsymbol{x}_1), \ldots, m(\boldsymbol{x}_n)$. We then use these values to calculate an estimate of P, which we denote by $\widehat{P} = \{\hat{p}_{ij}\}$, where $\hat{p}_{ij}$ is the proportion of iterations t with $m(\boldsymbol{x}_t) = i$ for which $m(\boldsymbol{x}_{t+1}) = j$.

When comparing the MCMC sampling methods, we shall focus on a few areas:

1. Mode jumping rate

   The "mode jumping rate" is defined to be the proportion of iterations t where $m(\boldsymbol{x}_t) \neq m(\boldsymbol{x}_{t+1})$, for $t = 1, \ldots, n-1$. The higher the mode jumping rate is, the better the method.

2. Number of function calls used

   We shall compare the number of calls to the function $\pi(\boldsymbol{x})$ used by each method. The smaller the number is, the better the method.

3. Rate of convergence for the whole chain

   We shall evaluate the rate of convergence of the process $\{m(\boldsymbol{X}_t)\}$ to its limiting distribution. To do this, we make the approximation that $\{m(\boldsymbol{X}_t)\}$ is Markov and analyze its eigenvalues, as explained in section 2.1.5. The rate of convergence is given by λ^*, the second largest in absolute value eigenvalue of $\widehat{P}$.

4. Accuracy of estimation

   We estimate the probability under π of each mode by estimating the mean of $f^{(i)}(\boldsymbol{X}) = I\{m(\boldsymbol{X}) = i\}$, for $i = 1, \ldots, m$, since $\mathbb{E}(I\{m(\boldsymbol{x}) = i\}) = \mathbb{P}(m(\boldsymbol{X}) = i)$.

Then we can compare the accuracy of these estimate under different methods by comparing values of the IAC $\tau(f^{(i)})$, as defined in section 2.1.6. The smaller the value of $\tau(f^{(i)})$ is, the faster the accuracy of the estimate increases with the number of iterations. This result should, therefore, be combined with information on the computational burden of each iteration.

We could estimate $\tau(f^{(i)})$ by doing repeated sampling of runs of length n, say N times, where for each run we calculate the proportion of time the chain spends in mode i, $\bar{f}^{(i)} = \sum_{t=1}^{n} f^{(i)}(\boldsymbol{x}_t)/n$. The N runs then give the values $\bar{f}_t^{(i)}$, $t = 1, \ldots, N$. We estimate the variance of the distribution of these $\bar{f}_t^{(i)}$ by using

$$\mathrm{Var}(\bar{f}^{(i)}) = \sum_{t=1}^{N} \frac{(\bar{f}_t^{(i)} - \bar{\bar{f}}^{(i)})^2}{N-1},$$

where $\bar{\bar{f}}^{(i)} = \sum_{t=1}^{N} \bar{f}_t^{(i)}/N$. Then we can estimate $\tau(f^{(i)})$ by using

$$\tau(f^{(i)}) = \frac{n}{\sigma_i^2}\mathrm{Var}(\bar{f}^{(i)}) \tag{4.2.1}$$

from equation (2.1.10) in section 2.1.6, where σ_i^2 is the variance of $f^{(i)}(\boldsymbol{x})$ at equilibrium. Since $\mathbb{E}(f^{(i)}(\boldsymbol{x})) = \mathbb{P}(\text{mode } i)$ and we can estimate $\mathbb{E}(f^{(i)}(\boldsymbol{x}))$ by $\bar{\bar{f}}^{(i)}$, we can estimate σ_i^2 by $\bar{\bar{f}}^{(i)}(1 - \bar{\bar{f}}^{(i)})$.

However, using repeated runs to estimate $\tau(f^{(i)})$ is computationally intensive. A less computationally intensive method of estimating $\tau(f^{(i)})$ is by using

$$\tau(f^{(i)}) = \frac{1}{\sigma^2 n} \sum_{s=1}^{n} \sum_{t=1}^{n} \mathrm{cov}[f^{(i)}(\boldsymbol{x}_s), f^{(i)}(\boldsymbol{x}_t)], \tag{4.2.2}$$

again from equation (2.1.10). Estimating $\tau(f^{(i)})$ using equation (4.2.2) is fairly tricky, as we need to truncate the sums and estimate $\mathrm{cov}[f^{(i)}(\boldsymbol{x}_s), f^{(i)}(\boldsymbol{x}_t)]$ with a long lag. However, we have a novel approach based on the estimated transition matrix $\widehat{P}$. Expanding equation (4.2.2), we obtain

$$\tau(f^{(i)}) = \frac{1}{\sigma^2 n}\left(n\,\mathrm{var}[f^{(i)}(\boldsymbol{x}_1)] + 2\sum_{k=1}^{n-1}(n-k)\,\mathrm{cov}[f^{(i)}(\boldsymbol{x}_1), f^{(i)}(\boldsymbol{x}_{k+1})] \right). \tag{4.2.3}$$

We shall then estimate terms in the right-hand side of equation (4.2.3) using $\widehat{P}$. Suppose there are m modes present in π. We denote the probability under π for each mode i by w_i, $i = 1, \ldots, m$, where $w_i = \mathbb{P}(m(\boldsymbol{X}) = i) = \mathbb{E}(f^{(i)})$. Let P^k denote the product of k copies of P, where $P^k = \{p_{ij}^k\}$. Under the assumption that the process $\{m(\boldsymbol{X}_t)\}$ is Markov, its transition matrix P has the properties:

(a) P^r will converge to a matrix with each row equal to $w_1, \ldots, w_m$ as r goes to infinity;

(b) $\text{var}[f^{(i)}] = w_i\,(1 - w_i)$;

(c) $w_i\,p_{ii}^k = \mathbb{P}\,(m(\boldsymbol{X}_s) = i \text{ and } m(\boldsymbol{X}_{s+k}) = i)$ when $\boldsymbol{X}_s \sim \pi$ initially; and

(d) $$\begin{aligned}\text{cov}[f^{(i)}(\boldsymbol{x}_1), f^{(i)}(\boldsymbol{x}_{k+1})] &= \mathbb{E}[f^{(i)}(\boldsymbol{x}_1), f^{(i)}(\boldsymbol{x}_{k+1})] - \mathbb{E}[f^{(i)}(\boldsymbol{x}_1)] \cdot \mathbb{E}[f^{(i)}(\boldsymbol{x}_{k+1})] \\ &= w_i\,p_{ii}^k - w_i^2.\end{aligned}$$

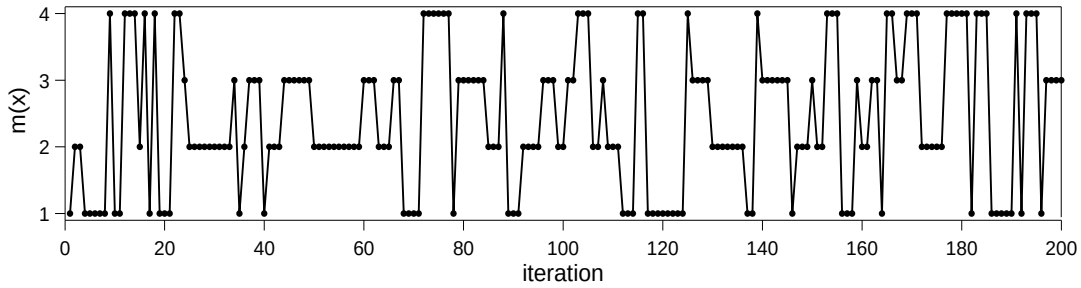
We have $\widehat{P}$ as an approximation to P. Therefore we take a row of $\widehat{P}^r$ for large r as an estimate of $(w_1, \ldots, w_m)$, which we denote by $\hat{\boldsymbol{w}}$. With this in place, $\hat{w}_i\,\hat{p}_{ii}^k - \hat{w}_i^2$ is an estimate of $\text{cov}[f^{(i)}(\boldsymbol{x}_1), f^{(i)}(\boldsymbol{x}_{k+1})]$.

There will be a small amount of error due to sampling error when estimating P by $\widehat{P}$ — small as we use very many iterations to calculate $\widehat{P}$. There is also additional error due to the fact that $\{m(\boldsymbol{X}_t)\}$ is not quite Markov. However, as long as this process is close to Markov, we have a very effective way of estimating the covariance terms, with no extra difficulty arising as the lag k increases. In contrast, methods based on direct estimation of covariances from data sequences are known to have difficulty when covariances continue up into high values of k.

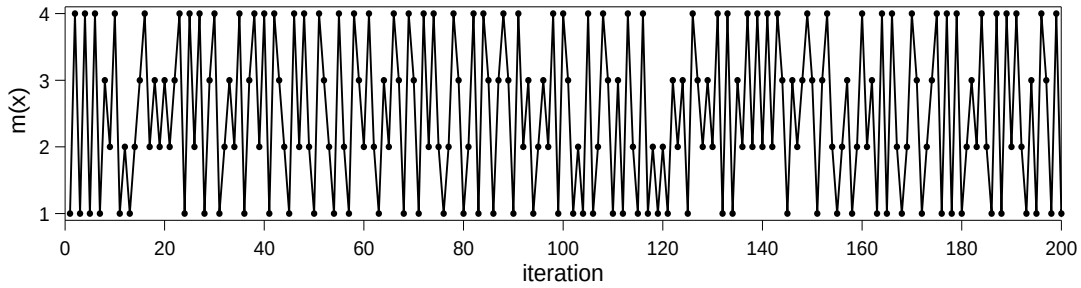
4.2.3 Results for Example 2

The initial exploration and clustering stages were implemented and four clusters were obtained, therefore m is equal to 4. The values of $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_4$ turned out to be very close to the true values (0, 0), (1, 0), (0, −1) and (1, −1) respectively, with errors smaller than 1.0×10^{-10}. Note that the same results were obtained for several different runs of the initial exploration and clustering stages; we shall discuss the initial exploration stage further in section 4.2.4. In the modelling stage, the approximate normal distributions fitted at each $\boldsymbol{\eta}_i$ were very close to the true distribution at that mode, and the approximate weights for each mode were also very close to their true values of 0.25.

In the sampling stage, we simulated a chain of length $n = 10,000$ using the Tjelmeland & Hegstad (2001) method (as implemented in section 3.2.3) and our MJM method. The values of $m(\boldsymbol{x})$ for the first 200 iterations are shown in Figure 4-2. From Figure 4-2, we can see that the chain simulated using our method is mixing better, i.e., there are more movements between the modes. We can also see this when we compare the estimate from these 10,000 iterations of the transition matrices for the movement of the chain between the four modes for these two methods, which are given by



(a) the Tjelmeland & Hegstad (2001) method



(b) the MJM method

Figure 4-2: Plots of the values of $m(\boldsymbol{x})$ for the first 200 iterations, where the chains are simulated using the Tjelmeland & Hegstad (2001) method and the MJM method.

$$\widehat{P}_{th} = \begin{pmatrix} 0.684 & 0.082 & 0.084 & 0.150 \\ 0.076 & 0.671 & 0.170 & 0.083 \\ 0.079 & 0.159 & 0.690 & 0.072 \\ 0.163 & 0.087 & 0.071 & 0.679 \end{pmatrix}$$

for the Tjelmeland & Hegstad (2001) method, and

$$\widehat{P}_{m} = \begin{pmatrix} 0.0 & 0.335 & 0.333 & 0.332 \\ 0.311 & 0.0 & 0.341 & 0.348 \\ 0.320 & 0.344 & 0.0 & 0.336 \\ 0.330 & 0.340 & 0.330 & 0.0 \end{pmatrix}$$

for the MJM method. We can see that there are larger values off the diagonal in $\widehat{P}_m$. The overall results for both methods are summarized in Table 4.1. We shall go through these results and discuss them one by one.

From Table 4.1, we can see that our MJM method yields a much higher mode jumping rate compared to Tjelmeland & Hegstad (2001) method. In fact, the MJM method has a 100% mode jumping rate. The reason for this is because in the modelling stage this method fitted a local bivariate normal at each mode to the true distribution which is a mixture of bivariate normal distributions. Furthermore, the quasi-Newton

	Method	
	T & H	MJM
Mode jumping rate (%)	31.90	100.00
Function calls per iteration		
a) initial exploration	0	32
b) sampling		
i) local steps	5	5
ii) mode jumping step	78	1
Total function calls ($\times 10^4$)		
a) initial exploration	0	0.32
b) sampling	83	6
Eigenvalue λ_k of $\widehat{P}$ for which $\lvert\lambda_k\rvert = \lambda^*$	0.683	-0.342
Estimated IAC for $f^{(i)} = I\{m(\boldsymbol{x}) = i\}$ calculated using repeated sampling		
1) $\hat{\tau}(f^{(1)})$	4.554	0.573
2) $\hat{\tau}(f^{(2)})$	4.112	0.449
3) $\hat{\tau}(f^{(3)})$	3.578	0.468
4) $\hat{\tau}(f^{(4)})$	4.096	0.503
Estimated IAC for $f^{(i)} = I\{m(\boldsymbol{x}) = i\}$ calculated using $\widehat{P}$		
1) $\hat{\tau}(f^{(1)})$	3.923	0.515
2) $\hat{\tau}(f^{(2)})$	3.711	0.493
3) $\hat{\tau}(f^{(3)})$	3.983	0.498
4) $\hat{\tau}(f^{(4)})$	3.915	0.494

Table 4.1: Overall results for a chain of run length 10,000, simulated using the Tjelmeland & Hegstad (2001) method and the MJM method.

optimization algorithm used in the initial exploration stage managed to get values very close to the real modes, which leads to the inverse of the approximate Hessian having values very close to the real covariance matrix. Hence, the fitted local bivariate normal is very close to the true distribution at that mode, which also means that the approximate weight w_i is also very close to the true weight ω_i. Since the true weight ω_i is the same for $i = 1, \ldots, 4$, this means that in the calculation of $\alpha(\boldsymbol{x}, \boldsymbol{y})$ the value of $\pi(\boldsymbol{y})\, p_{h(y),h(x)}\, g_{h(x)}(\boldsymbol{x})$ is very close to $\pi(\boldsymbol{x})\, p_{h(x),h(y)}\, g_j(\boldsymbol{y})$ and these cancel each other out, which results in an acceptance probability which is very close to one. Note that this will not happen if ω_i is not the same for $i = 1, \ldots, 4$.

From the values of the mode jumping rate given in Table 4.1, we can see that the mode jumping proposals in the MJM method will successfully jump between modes at least three times as frequently as the proposals in the Tjelmeland & Hegstad (2001) method. Furthermore, the Tjelmeland & Hegstad (2001) method uses 78 calls to the function $\pi(\boldsymbol{x})$ in each mode jumping step. This is because in each mode jumping step the Tjelmeland & Hegstad (2001) method has to hill-climb to the nearest mode location, fit a normal distribution there, and do these operations again on the reverse move. Our method only uses one call to the function $\pi(\boldsymbol{x})$ in each mode jumping step, which is used when evaluating $\alpha(\boldsymbol{x}, \boldsymbol{y})$.

The MJM method has a value of λ^* equal to 0.342. This is nearly half of the value of λ^* obtained by the Tjelmeland & Hegstad (2001) method, which is 0.683, therefore convergence to the correct distribution over the four modes is faster for the MJM method. Note that for the MJM method, the value of λ_k for which $|\lambda_k| = \lambda^*$ is negative. According to Green & Han (1992), rapid weak convergence to equilibrium is obtained by having eigenvalues, except for $\lambda_0 = 1$, which are small in absolute value, whilst good asymptotic mean squared error of estimation is helped by having negative eigenvalues. In our case, the value of λ_k for our method is both smaller in absolute value and negative, meaning that our method has faster convergence and is likely to produce better estimation. Note that the negative value of λ_k in our method can be attributed to the fact that our method has "forced negative correlation", i.e., our method always forces the mode jumping step to propose a new mode which is different from the current mode.

In section 4.2.2 we discussed different ways of estimating the value of $\tau(f^{(i)})$ for the function $f^{(i)}(\boldsymbol{x}) = I\{m(\boldsymbol{x}) = i\}$, $i = 1, \ldots, m$. For the first approach, we estimate the variance of the estimated probability associated with each mode by simulating $N = 100$ repeated runs of the chain, and estimate $\mathbb{E}(f^{(i)}(\boldsymbol{x}))$ by the proportion of time the chain spends in mode i. A histogram of the estimates for each mode is shown in Figure 4-3 for the Tjelmeland & Hegstad (2001) method and in Figure 4-4 for the

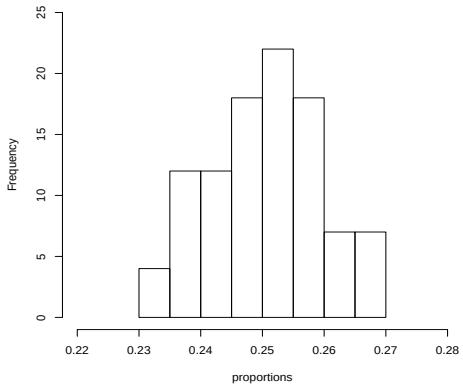
MJM method. From the figures, we can see that for both methods, the proportion of time the chain spends in each mode is not that far from the expected value of 0.25. However, our method produces estimates which are more accurate, as the values have smaller dispersion. We used these simulated proportions to estimate $\tau(f^{(i)})$ by using equation (4.2.1) from section 4.2.2. The corresponding values of $\hat{\tau}(f^{(i)})$ are given in Table 4.1. We can see that the values of $\hat{\tau}(f^{(i)})$ for the MJM method are around one eighth of the values for the Tjelmeland & Hegstad (2001) method, i.e., our method obtains the same estimation accuracy as the Tjelmeland & Hegstad (2001) method using just one eighth of the run length. This means that the MJM method has better estimation performance than the Tjelmeland & Hegstad (2001) method. The fact that our method has values of $\hat{\tau}(f^{(i)})$ which are less than one can be attributed to the forced negative correlation in our method.

Since the true weights at all the modes are equal and the distribution π and MCMC algorithms have symmetry, $\tau(f^{(i)})$ should be equal for $i = 1, \ldots, 4$. However, the values of $\hat{\tau}(f^{(i)})$ calculated using repeated sampling as given in Table 4.1 are fairly different from each other. Further analysis of the variance of the estimates of $\mathrm{Var}(\bar{f}^{(i)})$ shows that the estimated values of $\tau(f^{(i)})$ for the Tjelmeland & Hegstad (2001) method using this approach have a standard error of around 0.53, which explains the observed differences.

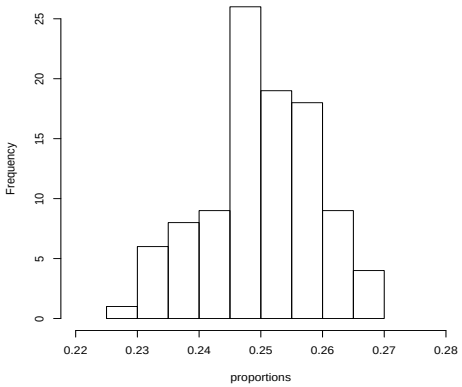
In the second approach, we estimate $\tau(f^{(i)})$ by using $\widehat{P}$. The corresponding values of $\hat{\tau}(f^{(i)})$ for this approach are given in Table 4.1. The values of $\hat{\tau}(f^{(i)})$ calculated using $\widehat{P}$ are closer to each other than the values obtained using repeating sampling, which suggests that the values of $\tau(f^{(i)})$ estimated using $\widehat{P}$ are more reliable. We can check this by using symmetry to adjust $\widehat{P}_{th}$ and $\widehat{P}_m$ to obtain a more accurate value of $\widehat{P}$ for the Tjelmeland & Hegstad (2001) method and the MJM method respectively. Looking at $\widehat{P}_{th}$, we would expect that there to be symmetry in the movements between modes 1 and 2, modes 1 and 3, modes 3 and 4 and modes 2 and 4, while there is a separate symmetry in the movements between modes 1 and 4 and modes 2 and 3. If we average out these two groups of values, we obtain

$$\widehat{P}'_{th} = \begin{pmatrix} 0.681 & 0.079 & 0.079 & 0.161 \\ 0.079 & 0.681 & 0.161 & 0.079 \\ 0.079 & 0.161 & 0.681 & 0.079 \\ 0.161 & 0.079 & 0.079 & 0.681 \end{pmatrix}.$$

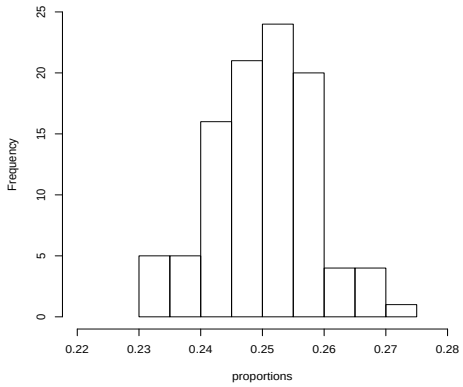
If we use $\widehat{P}'_{th}$ to calculate $\hat{\tau}(f^{(i)})$, we obtain $\hat{\tau}(f^{(i)}) = 3.872$ for all i. We can see that the values of $\hat{\tau}(f^{(i)})$ calculated using $\widehat{P}_{th}$ are fairly close to this value. Likewise, looking at $\widehat{P}_m$ we can see that there are symmetry for the values off the diagonal. If we average



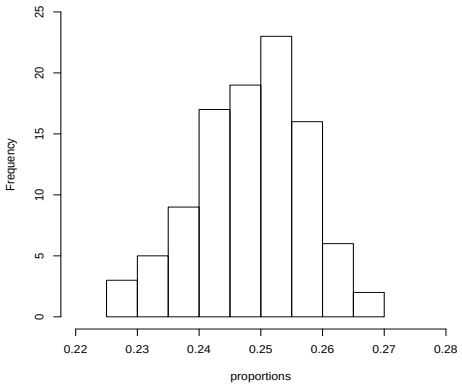
(a) Mode location = (0, 0)



(b) Mode location = (1, 0)

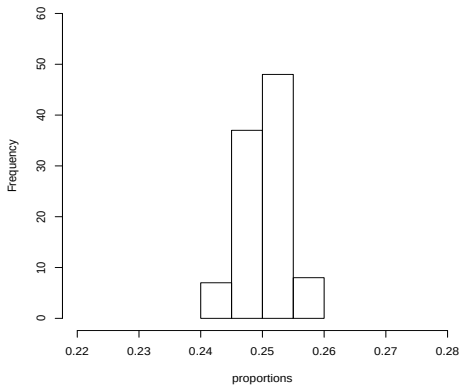


(c) Mode location = (0, −1)

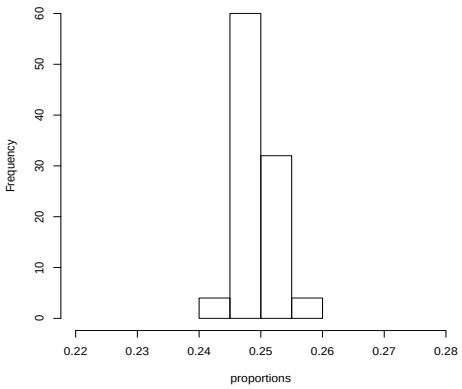


(d) Mode location = (1, −1)

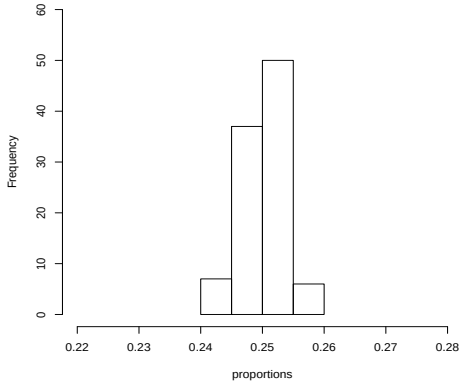
Figure 4-3: Histograms for the proportion of time the chain spends in each mode, where the chain is simulated using the Tjelmeland & Hegstad (2001) method.



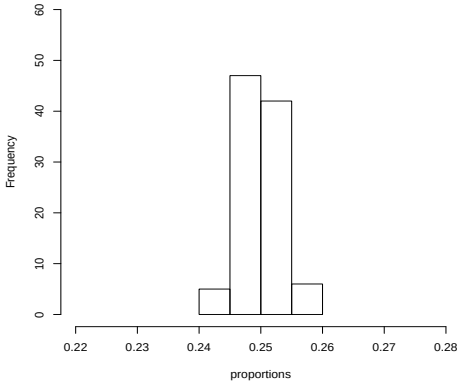
(a) Mode location = (0, 0)



(b) Mode location = (1, 0)



(c) Mode location = (0, −1)



(d) Mode location = (1, −1)

Figure 4-4: Histograms for the proportion of time the chain spends in each mode, where the chain is simulated using the MJM method.

out this group of values, we would obtain

$$\widehat{P}'_m = \begin{pmatrix} 0.000 & 0.333 & 0.333 & 0.333 \\ 0.333 & 0.000 & 0.333 & 0.333 \\ 0.333 & 0.333 & 0.000 & 0.333 \\ 0.333 & 0.333 & 0.333 & 0.000 \end{pmatrix}$$

If we use $\widehat{P}'_m$ to calculate $\hat{\tau}(f^{(i)})$, we obtain $\hat{\tau}(f^{(i)}) = 0.5$ for all i. We can see that the values of $\hat{\tau}(f^{(i)})$ calculated using $\widehat{P}_m$ are fairly close to this value. Based on these results we can conclude that the values of $\tau(f^{(i)})$ estimated using $\widehat{P}$ are more reliable than the values estimated using repeated sampling, and the differences between $\hat{\tau}(f^{(1)})$ to $\hat{\tau}(f^{(4)})$ calculated using $\widehat{P}_{th}$ and $\widehat{P}_m$ can be attributed to random variation, i.e., sampling error. Note that these results really depend on the Markov assumption for the $\{m(\boldsymbol{X}_t)\}$ process, and we shall prove in section 4.2.5 that there is no evidence against this assumption.

When we compare the value $\hat{\tau}(f^{(i)}) = 3.87$ for the Tjelmeland & Hegstad (2001) method and $\hat{\tau}(f^{(i)}) = 0.5$ for the MJM method, this means the MJM method obtains the same estimation accuracy as the Tjelmeland & Hegstad (2001) method using around one eighth of the run length. However, we also have to take into account the cost of each method. If we combine this with the fact that the MJM method uses less function calls per iteration in the mode jumping step compared to the Tjelmeland & Hegstad (2001) method, with the ratio of 1 : 78, then overall the MJM method is 600 times more efficient than the Tjelmeland & Hegstad (2001) method.

Although our mode jumping approach performs more efficiently than the Tjelmeland & Hegstad (2001) method in the sampling stage, our method does have an extra setup cost which needs to be considered in efficiency comparisons. In the initial exploration stage, our method uses on average 32 function calls for each of the 100 iterations. Since the total number of function calls used in the initial exploration stage is only 5% of the total number used in the sampling stage, this initial exploration is relatively cheap and accounting for it makes little difference to the efficiency comparisons noted above. However, we still have to consider the issue of failing to find a mode in the initial exploration which would, of course, be a serious drawback for our method. We build on the theoretical reassurances presented in section 3.4 by analyzing this example further in the next section.

4.2.4 Controlling the risk of missing a mode in the initial exploration

One point of concern, peculiar to our mode jumping approach, is the danger of failing to find a mode in the initial exploration stage. We shall address this directly for Example

2, by looking at the probability of missing a mode. Consider n short optimization runs, each starting from the final value of the previous run. We define the mode at the end of each short run as the state of a Markov chain M_t. We are interested in the probability of failing to find a mode, for example mode 4, at all in the n short runs when starting at some initial mode value M_0, which can be denoted by

$$\mathbb{P}(M_t \neq 4 \text{ for all } t = 1, \ldots, n | M_0 = j),$$

for $j = 1, \ldots, 3$. We can estimate this probability by using the transition matrix P_{in} for the process $\{M_t\}$. Since the true value of P_{in} is unknown, we shall estimate it by simulating a large number of short optimization runs. We shall denote this estimate of P_{in} by $\widehat{P}_{in}$.

We then define a new Markov chain with variable Y_t which has 7 states, where

$$Y_t = \begin{cases} 1, & \text{if } M_t = 1 \text{ and } M_i \neq 4,\ i = 0, \ldots, t-1 \\ 2, & \text{if } M_t = 1 \text{ and } M_i = 4 \text{ for some } i \in \{0, \ldots, t-1\} \\ 3, & \text{if } M_t = 2 \text{ and } M_i \neq 4,\ i = 0, \ldots, t-1 \\ 4, & \text{if } M_t = 2 \text{ and } m_i = 4 \text{ for some } i \in \{0, \ldots, t-1\} \\ \vdots & \\ 7, & \text{if } M_t = 4. \end{cases}$$

The corresponding transition matrix $\widehat{P}_Y$ for the process $\{Y_t\}$ can be calculated from $\widehat{P}_{in}$. Let $\widehat{P}_Y^n$ denote the product of n copies of $\widehat{P}_Y$. The probability of failing to find mode 4 in n short optimization runs when starting at mode j, $j \neq 4$, can be estimated by

$$\mathbb{P}(M_t \neq 4 \text{ for all } t = 1, \ldots, n | M_0 = j) \approx \widehat{P}_Y^n(j,1) + \widehat{P}_Y^n(j,3) + \widehat{P}_Y^n(j,5). \qquad (4.2.4)$$

We simulated one million short optimization runs to obtain

$$\widehat{P}_{in} = \begin{pmatrix} 0.357 & 0.242 & 0.241 & 0.160 \\ 0.240 & 0.360 & 0.159 & 0.241 \\ 0.239 & 0.162 & 0.361 & 0.238 \\ 0.161 & 0.241 & 0.241 & 0.357 \end{pmatrix}.$$

In the previous section we used $n = 100$ in the initial exploration stage of our mode jumping approach. For $n = 100$, calculation shows the probability of missing mode 4 when running the initial exploration stage of our mode jumping approach is approximately

$$\mathbb{P}(M_t \neq 4 \text{ for all } t = 1, \ldots, n | M_0 = j) \approx 4 \times 10^{-11},$$

for $j = 1, \ldots, 3$. Therefore, we can see that even with a small search process with just 100 runs the probability of missing mode 4 in the initial exploration stage of our mode jumping approach is very small. As a function of n, the probability of missing mode 4 when running the initial exploration stage of our mode jumping approach is approximately

$$0.9 \times 10^{-n/9.7}. \tag{4.2.5}$$

A initial exploration with a large number of short runs will then send the probability of missing a mode to zero.

We can draw comparisons with the probability of missing mode 4 in the full sampling run when Tjelmeland & Hegstad (2001) method is applied. Suppose $n = 10{,}000$ are performed, which uses 830,000 function calls. Note that estimates of w_i have variance

$$\frac{1}{4} \cdot \frac{3}{4} \cdot \frac{1}{10000/3.87} \approx 0.01^2,$$

which means we expect to see $\hat{w}_i = 0.25 \pm 0.02$. Using the transition matrix $\widehat{P}'_{th}$ from the previous section and the expression (4.2.4), the probability of missing mode 4 when sampling for 10,000 iterations of the Tjelmeland & Hegstad (2001) mode jumping method is estimated to be

$$\mathbb{P}(M_t \neq 4 \text{ for all } t = 1, \ldots, n | M_0 = j) \approx 1.3 \times 10^{-470},$$

for $j = 1, 2, 3$, which is tiny. One would not necessarily worry about obtaining such a small probability in the MJM method, but let us show that it can be done. From (4.2.5), we work out that we could achieve an equally small probability of missing mode 4 in the initial exploration stage of our method if we ran the initial exploration using 4,500 short runs instead of 100. This would increase the total number of function calls used in the initial exploration stage to 144,000. One short optimization run in our initial exploration stage actually takes less than half the function calls of a Tjelmeland & Hegstad (2001) mode jumping step. For the MJM method to obtain the same variance of estimation of w_i as the Tjelmeland & Hegstad (2001) method, it must have a sampling run of length

$$10,000 \times \frac{0.5}{3.87} \approx 1292,$$

which needs about 7,752 function calls. If we compare the total number of function calls used by the MJM method, which is 144,000 + 7,752, to the number used by the Tjelmeland & Hegstad (2001) method, which is 830,000, the MJM method then matches all aspects of the Tjelmeland & Hegstad (2001) method with just one fifth of the computation. We can also decrease $\text{var}(\hat{w}_i)$ a lot for not much more work. While running an initial exploration for so long to get such an extremely small probability of

Number of function calls	Probability of missing mode 4		Variance of estimates of w_i	
	MJ	T & H	MJ	T & H
10,000	10^{-7}	10^{-6}	0.0084^2	0.0774^2
100,000	10^{-64}	10^{-55}	0.0011^2	0.0245^2
1,000,000	10^{-644}	10^{-555}	0.0003^2	0.0078^2

Table 4.2: The results for our mode jumping method when using the ratio of 4:1 for the number of function calls used in sampling and running the initial exploration compared to those for the full sampling run of the Tjelmeland & Hegstad (2001) method, for various numbers of function calls.

missing a mode is not really necessary, it is still an option as it is still more efficient that using the Tjelmeland & Hegstad (2001) method. Since the sampling stage of our mode jumping method is very efficient, our method can afford the cost of running the initial exploration for any substantial amount of time.

However, instead of running an initial exploration for so long to get such an extremely small probability of missing a mode, we can use any sensible options. We have to find a balance between initial exploration and sampling. For example, consider 4:1 for the number of function calls in sampling and the initial exploration of the MJM method. Suppose we work with 10,000 function calls in total — 2,000 for initial exploration, which is equivalent to 63 iterations, and 8,000 for sampling, which is equivalent to 1,333 iterations. In this case the probability of failing to find mode 4 is approximately 10^{-7} and estimates of w_i have variance around 0.0084^2. On the other hand, sampling using 10,000 function calls using the Tjelmeland & Hegstad (2001) method is equivalent to using 121 iterations and produce probability of failing to find mode 4 which is approximately 10^{-6} and estimates of w_i with variance around 0.0774^2. We can see that in this case the probability of missing mode 4 and the variance of estimates of w_i for our method are lower than those for the Tjelmeland & Hegstad (2001) method. The results for the MJM method when using the ratio of 4:1 for the number of function calls used in sampling and running the initial exploration for different numbers of function calls, and the corresponding results for the Tjelmeland & Hegstad (2001) method, are given in Table 4.2. From this table we can see that for the given numbers of function calls, the probability of missing mode 4 and the variance of estimates of w_i for our method are always lower than those for the Tjelmeland & Hegstad (2001) method.

The high computational effort in the initial exploration reflects a "safety first" approach — we do not want to miss any mode at that stage. The return is that mode jumps are computationally cheap in the sampling stage, leading to accurate estimates in few iterations (compared to other types of mode jumping).

4.2.5 Testing the Markov assumption

In the previous sections, we have assumed that the process $\{m(\boldsymbol{X}_t)\}$ is approximately Markov; we would like to test this assumption. In order to show that the process $\{m(\boldsymbol{X}_t)\}$ is Markov, we have to show that the value of $m(\boldsymbol{X}_t)$ only depends on $m(\boldsymbol{X}_{t-1})$ and does not depend on $m(\boldsymbol{X}_{t-2}), \ldots, m(\boldsymbol{X}_0)$. However, it is difficult to test this directly. Instead, we shall test the more relaxed null hypothesis that the value of $m(\boldsymbol{X}_t)$ only depends on $m(\boldsymbol{X}_{t-1})$ and does not depend on $m(\boldsymbol{X}_{t-2})$ versus the alternative hypothesis that $m(\boldsymbol{X}_t)$ depends on both $m(\boldsymbol{X}_{t-1})$ and $m(\boldsymbol{X}_{t-2})$. If there is no significant evidence against this null hypothesis, then this is a positive indication regarding the validity of the Markov assumption for the $\{m(\boldsymbol{X}_t)\}$ process.

We shall define a matrix $Q = \{q_{ij,k}\}$ related to the process $\{m(\boldsymbol{X}_t)\}$, where

$$q_{ij,k} \quad = \quad \mathbb{P}(m(\boldsymbol{X}_{t+1}) = k | m(\boldsymbol{X}_{t-1}) = i, m(\boldsymbol{X}_t) = j).$$

If the null hypothesis is true, then for any pair of j and k the values of $q_{ij,k}$ will be the same for all values of i. We can estimate Q by running the MCMC sampler for n number of iterations, where n is large, to obtain the sample values $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ and determine the corresponding values $m(\boldsymbol{x}_1), \ldots, m(\boldsymbol{x}_n)$. We then use these values to calculate an estimate of Q, which we denote by $\widehat{Q} = \{\hat{q}_{ij,k}\}$.

The values of $\widehat{Q}$ for 1 million iterations of the Tjelmeland & Hegstad (2001) method and the MJM method are given in Table 4.3. From this table we can see that for both methods, for any pair of j and k the values of $\hat{q}_{ij,k}$ for all i are similar and differences are in keeping with the standard deviations of approximately 0.002. Therefore, we can conclude that there is no evidence against the null hypothesis that the value of $m(\boldsymbol{X}_t)$ only depends on $m(\boldsymbol{X}_{t-1})$ and does not depend on $m(\boldsymbol{X}_{t-2})$. This then gives a positive indication regarding the validity of the Markov assumption for the $\{m(\boldsymbol{X}_t)\}$ process.

4.3 Mode jumping using differences

4.3.1 General methodology

We can also apply our other form of mode jumping, the MJD method, to sample from a target distribution with continuous variables in $\mathbb{R}^p$. To do this, we shall go through the three stages:

1. Initial exploration, where we try to find all possible locations of the modes.

2. Clustering, where we cluster the mode locations and find a single mode location to represent each cluster.

		T & H				MJM			
		k				k			
i	j	1	2	3	4	1	2	3	4
1	1	0.681	0.079	0.080	0.160	0.000	0.000	0.000	0.000
2		0.681	0.081	0.081	0.158	0.000	0.335	0.331	0.334
3		0.681	0.077	0.080	0.162	0.000	0.337	0.332	0.331
4		0.681	0.079	0.077	0.164	0.000	0.335	0.334	0.331
1	2	0.080	0.679	0.160	0.082	0.333	0.000	0.336	0.332
2		0.079	0.682	0.161	0.079	0.000	0.000	0.000	0.000
3		0.080	0.684	0.157	0.080	0.331	0.000	0.334	0.335
4		0.077	0.682	0.160	0.081	0.332	0.000	0.332	0.336
1	3	0.079	0.161	0.680	0.080	0.333	0.335	0.000	0.332
2		0.079	0.159	0.683	0.079	0.333	0.336	0.000	0.331
3		0.079	0.161	0.681	0.080	0.000	0.000	0.000	0.000
4		0.079	0.157	0.684	0.081	0.333	0.332	0.000	0.335
1	4	0.157	0.080	0.078	0.685	0.331	0.335	0.334	0.000
2		0.162	0.077	0.081	0.680	0.336	0.331	0.333	0.000
3		0.164	0.076	0.081	0.680	0.334	0.334	0.332	0.000
4		0.160	0.079	0.079	0.682	0.000	0.000	0.000	0.000

Table 4.3: The values of $\widehat{Q}$ for the 1 million iterations of the Tjelmeland & Hegstad (2001) method and the MJM method.

3. Sampling, where we use the differences between the mode locations to jump between the modes.

The initial exploration and clustering stages are the same as the ones in the MJM method, as described in section 4.1, while the sampling stage follows as in section 3.3.4. For the sampling stage we define $\boldsymbol{d}_{i,\,j} = \boldsymbol{\eta}_j - \boldsymbol{\eta}_i$ and $\boldsymbol{d}_{j,\,i} = -\boldsymbol{d}_{i,\,j}$. We take the operator $\oplus$ to be simple addition so that $\boldsymbol{y}' = \boldsymbol{x} \oplus \boldsymbol{d}_{h(x),j} = \boldsymbol{x} + \boldsymbol{d}_{h(x),j}$. We set $\boldsymbol{y} = \boldsymbol{y}' + \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathrm{N}_p(\boldsymbol{0}, \Sigma)$ with a predefined variance matrix Σ, where the elements of Σ are small. With these choices $q(\boldsymbol{y}', \boldsymbol{y}) = \mathrm{N}_p(\boldsymbol{0}, \Sigma)(\boldsymbol{y} - \boldsymbol{y}')$.

4.3.2 Application to Example 2

We shall apply the MJD method to sample from the mixture of four Gaussian distributions defined by equation (3.2.1) in section 3.2.2. Implementation of the initial exploration and clustering stages follows as in section 4.2.1. However, in the sampling stage we shall use the MJD method instead. We have applied this method in the special case where $\boldsymbol{\epsilon} = \boldsymbol{0}$ so that $\boldsymbol{y} = \boldsymbol{y}'$ and correspondingly $q(\boldsymbol{y}', \boldsymbol{y}) = 1$. Here we choose the probability of moving between mode i and j, p_{ij}, to be equal to $1/(m-1)$ for all i and j, $j \neq i$. Note that to make the MJD method comparable to the MJM method, we have chosen for the MJD method the same number of local changes and mode jumping

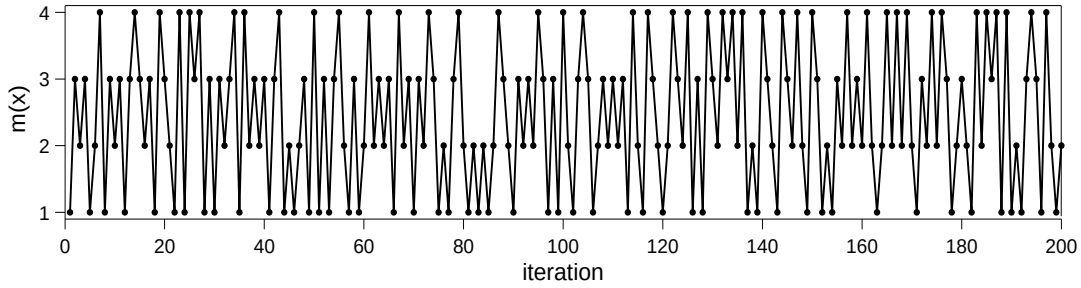


Figure 4-5: Plot of the values of $m(\boldsymbol{x})$ for the first 200 iterations, where the chain is simulated using the MJD method.

step per iteration and the same local proposal distribution as the ones in the MJM method, as given in section 4.2.1.

4.3.3 Results for example 2

The results for the initial exploration and clustering stages are the same as when we apply the MJM method (see section 4.2.3). In short, $m = 4$ clusters are obtained, where $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_4$ turned out to be very close to the true mode locations. In the sampling stage, we simulate a chain of length 10,000 using the MJD method. The values of $m(\boldsymbol{x})$ for the first 200 iterations are shown in Figure 4-5. From the figure, we can see that the chain simulated using this method is mixing well. The estimate from these 10,000 iterations of the transition matrix for the movement of the chain between the four modes is given by

$$\widehat{P}_d = \begin{pmatrix} 0.0 & 0.348 & 0.317 & 0.335 \\ 0.328 & 0.0 & 0.350 & 0.322 \\ 0.338 & 0.336 & 0.0 & 0.326 \\ 0.335 & 0.324 & 0.341 & 0.0 \end{pmatrix},$$

which is similar to the transition matrix when the chain is simulated using the MJM method (see $\widehat{P}_m$ in section 4.2.3).

The overall results for the MJD method are summarized in Table 4.4. We can see that the performance of this method is very similar to that of the MJM method, as given in Table 4.1. In fact, the MJD method also has a 100% mode jumping rate. The reason for this is because in the calculation of $\alpha(\boldsymbol{x}, \boldsymbol{y})$ the value of $\pi(\boldsymbol{y})$ is very close to $\pi(\boldsymbol{x})$ since the modes have the same local normal distributions and equal weights, and these cancel each other out, which results in an acceptance probability which is very close to one. One reason why the performances of the MJD method and the MJM method are so similar is that the target distribution in this example is a mixture of normal distributions which have the same weight and shape at every mode. The MJD method may not work as well if the shape at the modes are very different from each

Mode jumping rate (%)	100.00
Function calls per iteration	
a) initial exploration	32
b) sampling	
i) local steps	5
ii) mode jumping step	1
Total function calls ($\times 10^4$)	
a) initial exploration	0.32
b) sampling	6
Eigenvalue λ_k of $\widehat{P}$ for which	
$\lvert\lambda_k\rvert = \lambda^*$	-0.339
Estimated IAC for $f^{(i)} = I\{m(\boldsymbol{x}) = i\}$	
calculated using $\widehat{P}$	
1) $\hat{\tau}(f^{(1)})$	0.499
2) $\hat{\tau}(f^{(2)})$	0.497
3) $\hat{\tau}(f^{(3)})$	0.497
4) $\hat{\tau}(f^{(4)})$	0.506

Table 4.4: Overall results for a chain of run length 10,000, simulated using the MJD method.

other.

4.4 Application to an alternative example

We have applied our mode jumping approach to sample from the target distribution $\pi(\boldsymbol{x})$ defined by equation (3.2.1) in section 3.2.2, and we have shown that our mode jumping methods, i.e., the MJM method and the MJD method, work very well for this example. However, the target distribution defined by equation (3.2.1) is fairly simplistic as it is a mixture of normal distributions with equal weights and the same shape at each mode. We shall make $\pi(\boldsymbol{x})$ harder to sample from by varying the shape of the local distribution at each mode; we shall then apply our mode jumping approach to sample from this $\pi(\boldsymbol{x})$ and observe how well our approach performs.

4.4.1 Example 3: mixture of non Gaussian distributions

Suppose the target distribution is a mixture of four distributions on $\mathbb{R}^2$,

$$\pi(\boldsymbol{x}) = \sum_{i=1}^{4} \omega_i f_i(\boldsymbol{x}),$$

where $\omega_i = 1/4$ for $i = 1, \ldots, 4$. Let

$$f_i(\boldsymbol{x}) = \mathrm{N}_2(\boldsymbol{\mu}_i, \Sigma_i)(\boldsymbol{x})$$

for $i = 1, 2$, where $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ are $(0, 0)$ and $(1, 0)$ respectively, and the covariance matrices Σ_1 and Σ_2 are

$$\Sigma_1 = \begin{pmatrix} 0.01^2 & 0.9 \times 0.01^2 \\ 0.9 \times 0.01^2 & 0.01^2 \end{pmatrix},$$

and

$$\Sigma_2 = \begin{pmatrix} 0.01^2 & -0.9 \times 0.01^2 \\ -0.9 \times 0.01^2 & 0.01^2 \end{pmatrix}.$$

We shall define $f_i(\boldsymbol{x})$ for $i = 3, 4$ so that these density functions are transformations of the normal density function. Let $\boldsymbol{Z} \sim \mathrm{N}_2(\boldsymbol{\mu}_3, \Sigma_3)$, where $\boldsymbol{\mu}_3 = (0, -1)$ and $\Sigma_3 = \Sigma_1$. We define a transformation in the form of

$$\boldsymbol{X} = g(\boldsymbol{Z}) = (Z^{(1)} + 60\,(Z^{(2)} - \mu_3^{(2)})^2,\, Z^{(2)}).$$

The effect of this transformation is shown in Figure 4-6. This variable has density

$$\begin{aligned} f_3(\boldsymbol{x}) &= \mathrm{N}_2(\boldsymbol{\mu}_3, \Sigma_3)(g^{-1}(\boldsymbol{x})) \\ &= \mathrm{N}_2(\boldsymbol{\mu}_3, \Sigma_3)((x^{(1)} - 60\,(x^{(2)} - \mu_3^{(2)})^2,\, x^{(2)})), \end{aligned}$$

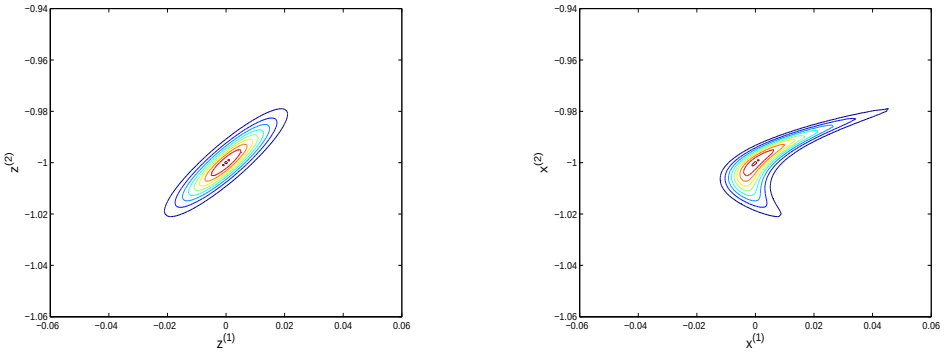
as the Jacobian of this transformation is equal to one. Similarly, we shall define

$$f_4(\boldsymbol{x}) = \mathrm{N}_2(\boldsymbol{\mu}_4, \Sigma_4)((x^{(1)} + 60\,(x^{(2)} - \mu_4^{(2)})^2,\, x^{(2)})),$$

where $\boldsymbol{\mu}_4 = (1, -1)$ and $\Sigma_4 = \Sigma_2$. Let $\rho_i = \Sigma_i^{(12)}/(\Sigma_i^{(11)}\Sigma_i^{(22)})^{1/2}$. The final target distribution $\pi(\boldsymbol{x})$ is then given by

$$\begin{aligned} \pi(\boldsymbol{x}) &= \sum_{i=1}^{4} \frac{\omega_i}{2\pi\sqrt{\Sigma_i^{(11)}\Sigma_i^{(22)}(1-\rho_i^2)}} \\ &\quad \times \exp\left\{ -\frac{1}{2(1-\rho_i^2)} \left[\left(\frac{x^{(1)} - a_i(x^{(2)} - \mu_i^{(2)})^2 - \mu^{(1)}}{\sqrt{\Sigma_i^{(11)}}} \right)^2 + \left(\frac{x^{(2)} - \mu^{(2)}}{\sqrt{\Sigma_i^{(22)}}} \right)^2 \right.\right. \\ &\quad \left.\left. -2\rho_i \left(\frac{x^{(1)} - a_i(x^{(2)} - \mu_i^{(2)})^2 - \mu^{(1)}}{\sqrt{\Sigma_i^{(11)}}} \right) \left(\frac{x^{(2)} - \mu^{(2)}}{\sqrt{\Sigma_i^{(22)}}} \right) \right] \right\} \end{aligned}$$

where $a_1, \ldots, a_4$ are 0, 0, 60, and -60 respectively; this full target distribution is shown in Figure 4-7.



(a) Before transformation (b) After transformation

Figure 4-6: The effect of applying the transformation $\boldsymbol{X} = g(\boldsymbol{Z}) = (Z^{(1)} + 60\,(Z^{(2)} - \mu_3^{(2)})^2,\, Z^{(2)})$ when $\boldsymbol{Z} \sim \mathrm{N}_2(\boldsymbol{\mu}_3, \Sigma_3)$.

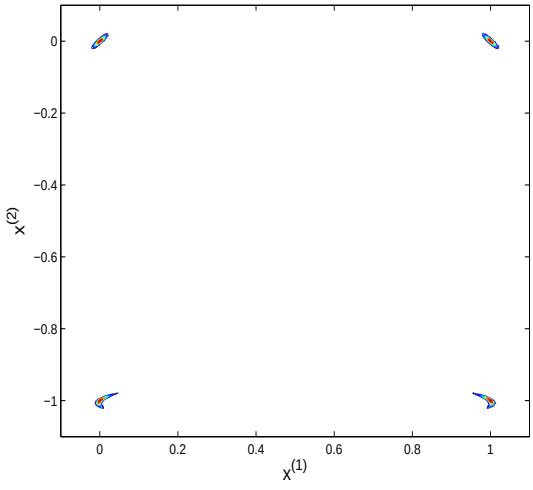


Figure 4-7: The true distribution for a mixture of non Gaussian distributions on all of $\mathbb{R}^2$.

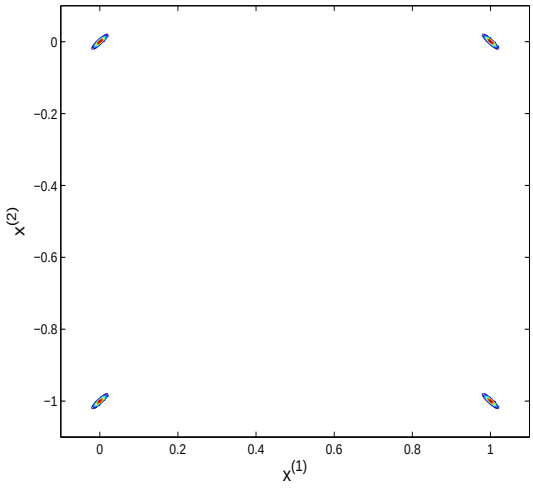


Figure 4-8: The approximate normal distributions fitted at $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_4$.

4.4.2 Implementation of the mode jumping approach

Implementation of the MJM method and the MJD method are the same as described in section 4.2.1 and section 4.3.2 respectively. However, in the sampling stage of both methods we choose to do local changes via 20 steps of the usual Metropolis-Hastings sampling method instead of 5. This is because we know that local steps are now necessary to explore each mode since the local distribution at some of the modes are no longer normal.

4.4.3 Results for Example 3

In the clustering stage, $m = 4$ clusters were formed. The values of $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_4$ turn out to be very close to the true values $(0, 0)$, $(1, 0)$, $(0, -1)$ and $(1, -1)$ respectively, with errors smaller than 1.0×10^{-10}. The approximate normal distributions fitted at $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_4$ in the modelling stage are shown in Figure 4-8. From this figure we can see that the distributions fitted at $\boldsymbol{\eta}_1 \approx (0, 0)$ and $\boldsymbol{\eta}_2 \approx (1, 0)$ are very close to the true distribution at these modes while those fitted at $\boldsymbol{\eta}_3 \approx (0, -1)$ and $\boldsymbol{\eta}_4 \approx (1, -1)$ are not; this is shown more clearly in Figure 4-9 for the case of mode 3. The approximate weights for the modes, $w_1, \ldots, w_4$, are calculated to be 0.2496, 0.2496, 0.2504 and 0.2504 respectively.

In the sampling stage, we simulate a chain of length 100,000 using the MJM method and the MJD method. The values of $m(\boldsymbol{x})$ for the first 200 iterations are shown in Figure 4-10. From the figure, we can see that the chains simulated using both methods are mixing fairly well, but the chain simulated using the MJM method is mixing better. We can also see this when we compare the estimate from these 100,000 iterations of the transition matrices for the movement of the chain between the four modes for both

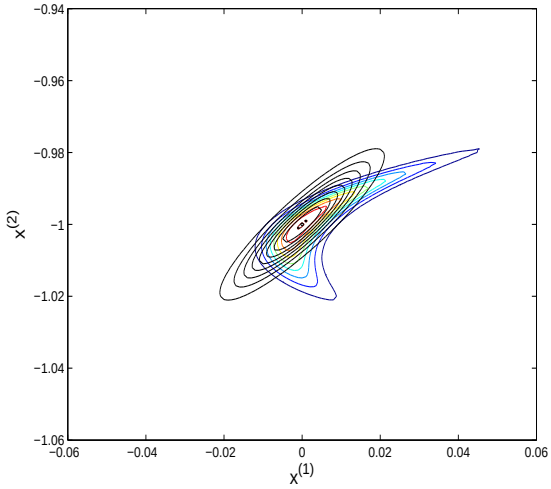


Figure 4-9: The true distribution at mode 3, with the fitted distribution in black.

methods, which are given by

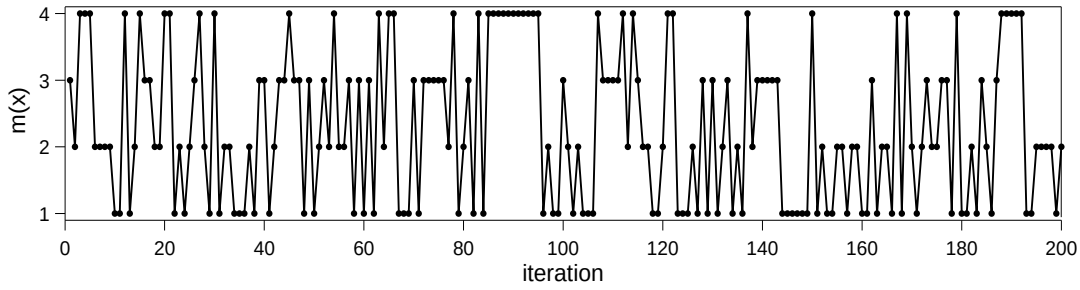
$$\widehat{P}_m = \begin{pmatrix} 0.240 & 0.333 & 0.215 & 0.212 \\ 0.334 & 0.241 & 0.212 & 0.213 \\ 0.218 & 0.211 & 0.429 & 0.142 \\ 0.210 & 0.213 & 0.152 & 0.425 \end{pmatrix}$$

for the MJM method, and

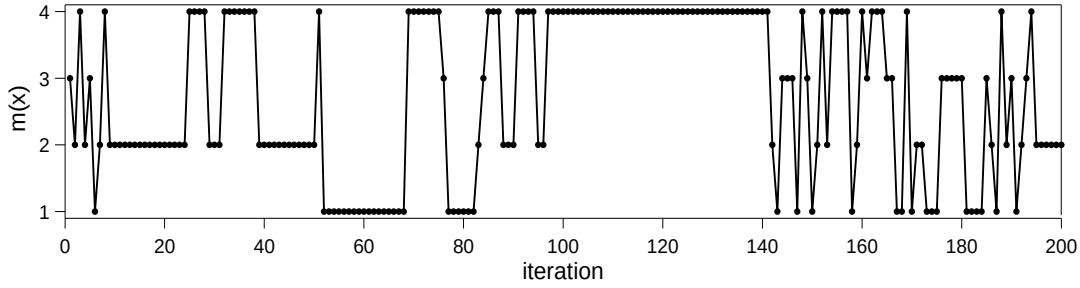
$$\widehat{P}_d = \begin{pmatrix} 0.594 & 0.094 & 0.207 & 0.105 \\ 0.090 & 0.599 & 0.103 & 0.208 \\ 0.205 & 0.106 & 0.551 & 0.138 \\ 0.106 & 0.213 & 0.133 & 0.549 \end{pmatrix}$$

for the MJD method. From these transition matrices we can see that there are larger values off the diagonal in $\widehat{P}_m$. For the MJM method, from its transition matrix $\widehat{P}_m$ we can see that this method has more problems jumping to and from modes 3 and 4. This is to be expected since this method is fitting approximate normal distributions at modes 3 and 4 when the distribution at these modes are not normal. For the MJD method, from its transition matrix $\widehat{P}_d$ we can see that this method is better at jumping between modes when the shapes at the modes are similar, for example it jumps more between modes 1 and 3, and between modes 2 and 4.

The overall results for both methods are summarized in Table 4.5. We can see that although both methods perform fairly well, the performance the MJM method is better than the MJD method as the MJM method has a higher mode jumping rate, faster convergence, i.e., smaller value of λ^*, and better estimation performance, i.e., smaller



(a) the MJM method



(b) the MJD method

Figure 4-10: Plots of the values of $m(\boldsymbol{x})$ for the first 200 iterations, where the chains are simulated using the MJM method and the MJD method.

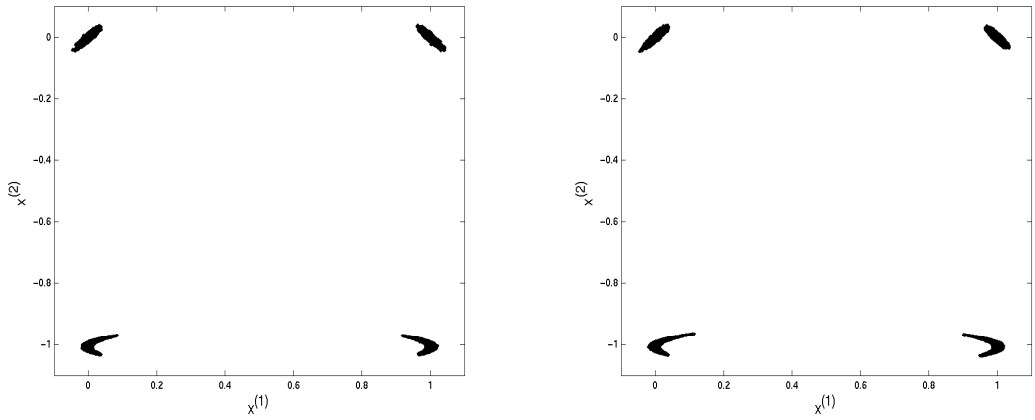
value of $\hat{\tau}(f^{(i)})$. These results confirm the fact that sampling from a target distribution $\pi(\boldsymbol{x})$ with modes of varying shapes is more difficult for the MJD method. Nevertheless, both methods seem fairly robust.

Note that even though $\pi(\boldsymbol{x})$ has been modified so that the local distributions at two of the modes are no longer normal, the samples from both methods manage to approximate the true distribution fairly well, as shown in Figure 4-11. The reason for this is the local movements. Even if the mode jumping step in our mode jumping methods fails to visit the tails of the distribution, the simulated MCMC chain can still visit there via the local movements; the suitable number of local movements that should be used will however depend on the application.

The Tjelmeland & Hegstad (2001) mode jumping method was not applied to this example because it has already been shown to be less efficient than our approach in the previous simpler example. If it were applied in this example we would expect it to experience problems with its reverse jumps due to the different basins of attraction of each mode.

	Method	
	MJM	MJD
Mode jumping rate (%)	66.65	42.69
Function calls per iteration		
a) initial exploration	59	59
b) sampling		
i) local steps	20	20
ii) mode jumping step	1	1
Total function calls ($\times 10^5$)		
a) initial exploration	0.059	0.059
b) sampling	21.0	21.0
Eigenvalue λ_k of $\widehat{P}$ for which $\lvert\lambda_k\rvert = \lambda^*$	0.280	0.572
Estimated IAC for $f^{(i)} = I\{m(\boldsymbol{x}) = i\}$ calculated using $\widehat{P}$		
1) $\hat{\tau}(f^{(1)})$	0.998	2.857
2) $\hat{\tau}(f^{(2)})$	1.004	2.859
3) $\hat{\tau}(f^{(3)})$	1.633	2.444
4) $\hat{\tau}(f^{(4)})$	1.634	2.406

Table 4.5: Overall results for a chain of run length 100,000, simulated using the MJM method and the MJD method.



(a) the MJM method (b) the MJD method

Figure 4-11: Plots of the values of $\boldsymbol{X}$ for a chain of length 100,000, where the chains are simulated using the MJM method and the MJD method.

4.5 Summary

Our mode jumping approach that uses information about the mode locations to jump between the modes effectively has been shown to perform much better than the Tjelmeland & Hegstad (2001) mode jumping method when applied to a given target distribution, as it is more efficient and gives more accurate estimates. We have also shown that our mode jumping approach is applicable even if the target distribution has modes with varying shapes, i.e., even if the target distribution is not normal.

Chapter 5

Application II: Discrete Variables

In this chapter, we consider an application in statistical image analysis which leads to a sampling problem where the target distribution has discrete variables. We shall apply our mode jumping approach to this problem and assess its performance.

5.1 The image restoration problem

Suppose we want to estimate a binary image from a signal recorded subject to additive noise. The image X and recorded signal Y are defined on a $p \times p$ array of pixels. It is convenient to index X and Y by a single label i and we write $X = \{X^{(i)}\}$ and $Y = \{Y^{(i)}\}$ where labels $i = 1, \ldots, p^2$ number the pixels in the array in a particular order, say row by row from top to bottom of the image. Then $X = \{X^{(i)}\}$, where $X^{(i)}$ is the discrete value at the i^{th} pixel, and $X^{(i)} \in \{a, b\}$ for $i = 1, \ldots, p^2$. The true image, X, is not directly observable, and we observe a noisy image $Y = \{Y^{(i)}\}$, where $P_{Y|X}(y|x)$ defines the density of the recorded signal at y. Here we assume that the $Y^{(i)}$ are conditionally independent given X. We shall also assume that $Y^{(i)}$ depends on X just through $X^{(i)}$.

In a Bayesian approach, we place a prior distribution $P_X(x)$ on X and estimate the true image from its posterior distribution, $P_{X|Y}(x|y)$, where

$$P_{X|Y}(x|y) \quad \propto \quad P_{Y|X}(y|x)\, P_X(x), \tag{5.1.1}$$

from Bayes theorem. Then in this problem, we shall define the target distribution $\pi(x)$ to be

$$\pi(x) = c\, P_{Y|X}(y|x)\, P_X(x),$$

for the appropriate value of c to make this a probability distribution, and we aim to sample from $\pi(x)$. We can do this by using the Gibbs sampler (Geman & Geman (1984)). To sample using the Gibbs sampler, we have to find the conditional posterior

distribution of each $X^{(i)}$ given Y and the other elements of X.

In the case of binary images, the prior $P_X(x)$ can be modelled by a locally dependent Markov random field (MRF) model (Besag (1974)). Such models are based on a definition of pairs of neighbouring pixels. Let ∂_i denote the neighbourhood of pixel i, where $\partial_i = \{j : j \text{ is a neighbour to } i\}$ and $X^{(\partial_i)} = \{X^{(j)} : j \in \partial_i\}$, and let $X^{(-i)} = \{X^{(j)} : j \neq i\}$. By the definition of a MRF with this neighbourhood,

$$P_{X^{(i)}|X^{(-i)}}(x^{(i)}|x^{(-i)}) = P_{X^{(i)}|X^{(\partial_i)}}(x^{(i)}|x^{(\partial_i)}). \tag{5.1.2}$$

This means that the value of $X^{(i)}$ is conditionally independent of $\{X^{(j)} : j \neq i, j \notin \partial_i\}$ given $X^{(\partial_i)}$. Using equation (5.1.2) and the fact that $P_{Y^{(i)}|X}(y^{(i)}|x) = P_{Y^{(i)}|X^{(i)}}(y^{(i)}|x^{(i)})$, we can reduce the conditional posterior distribution of $X^{(i)}$, $P_{X^{(i)}|Y,X^{(-i)}}(x^{(i)}|y, x^{(i)})$, to be

$$\begin{aligned} P_{X^{(i)}|Y,X^{(-i)}}(x^{(i)}|y, x^{(-i)}) &\propto P_{Y^{(i)}|X}(y^{(i)}|x)\, P_{X^{(i)}|X^{(-i)}}(x^{(i)}|x^{(-i)}) \\ &\propto P_{Y^{(i)}|X^{(i)}}(y^{(i)}|x^{(i)})\, P_{X^{(i)}|X^{(\partial_i)}}(x^{(i)}|x^{(\partial_i)}), \end{aligned} \tag{5.1.3}$$

for $i = 1, \ldots, p^2$. Then the conditional probability of $X^{(i)}$ given Y and $X^{(-i)}$ only depends on the value of the data $Y^{(i)}$ and its neighbours $\{X^{(\partial_i)}\}$.

The Gibbs sampler will use the conditional posterior distribution (5.1.3) to select a new state a at pixel i with probability $\mathbb{P}(X^{(i)} = a|Y = y, X^{(-i)} = x^{(-i)})$, or b with probability $1 - \mathbb{P}(X^{(i)} = a|Y = y, X^{(-i)} = x^{(-i)})$. When each $X^{(i)}$ has been updated in turn, for $i = 1, \ldots, p^2$, then a single cycle of the Gibbs sampler is completed, as is one step of the Markov chain. (For further information, see Geman & Geman (1984)). After ensuring that the chain has reached equilibrium, we can repeat the cycle for a large number of times to get a sample from $\pi(x)$, which we can then make use of to make inferences about $\pi(x)$ empirically. However, note that mixing may be slow if there are modes in $\pi(x)$, for example features made up of a number of pixels which may be present or absent. Then, "mode jumping" may be a useful addition to the Gibbs sampler.

5.2 Example 4: Location of archaeological sites

We shall look at a problem which involves finding the location of archaeological sites, using the data set from Besag et al. (1991). The data y, as shown in Figure 5-1, consist of a 16×16 grid of measurements of log phosphate level (refer to Appendix A for the actual values). The aim is to classify the region into areas with high and low phosphate levels, where high phosphate level indicates the presence of previous human activity and low phosphate level indicates the absence of it. This can be considered

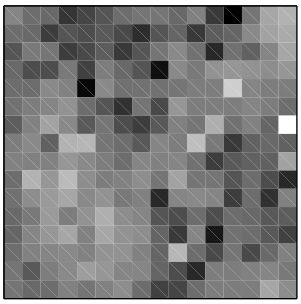


Figure 5-1: The log phosphate level data, where the values lie between low phosphate level (black) and high phosphate level (white).

as a problem in the restoration of a binary image, and has been discussed as such by Besag et al. (1991) and Gray (1994).

As in Besag et al. (1991) and Gray (1994) we take $Y^{(i)}$ as conditionally independent given X, and Gaussian with mean $X^{(i)}$, where $X^{(i)} = 1$ (absence) or 2 (presence of activity), and variance σ^2, so that $Y^{(i)}|X \sim \mathrm{N}(X^{(i)}, \sigma^2)$, for $i = 1, \ldots, 16^2$. The prior $P_X(x)$ is chosen to be the Potts model, for which

$$P(x) \propto \exp[-\beta V(x)],$$

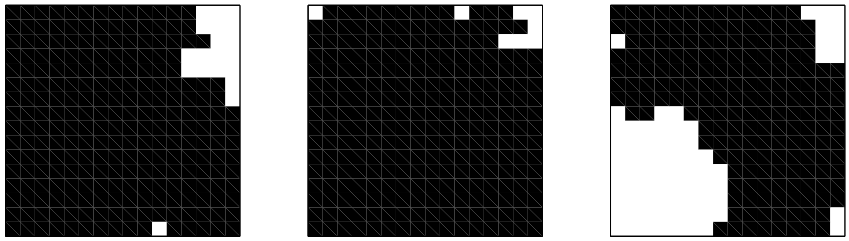
where β is a parameter that measures the strength of the interaction between neighbouring pixels, and

$$V(x) = \sum_i \sum_{j>i} I(j \in \partial_i \text{ and } X^{(j)} \neq X^{(i)}),$$

where in this case ∂_i is made up of pixels which are horizontal, vertical and diagonal neighbours to pixel i. Then the posterior distribution for X is given by

$$\pi(x) = c \exp\left\{ -\left[\frac{1}{2\sigma^2} \sum_{i=1}^{p^2} (y^{(i)} - x^{(i)})^2 \right] - \beta V(x) \right\}, \tag{5.2.1}$$

for the appropriate value of c to make this a probability distribution. Our task in MCMC simulation is to sample from π. In her example, Gray (1994) used the values $\beta = 0.738$ and $\sigma^2 = 0.434$. We shall start by using parameter values $\beta = 0.78$ and $\sigma^2 = 0.5$ which are plausible for the observed data and create some significant challenges for MCMC sampling.



(a) $t = 10{,}000$ (b) $t = 20{,}000$ (c) $t = 30{,}000$

Figure 5-2: Values of X at iterations t of the Gibbs sampler, where pixel i is black if $X^{(i)} = 1$ and white if $X^{(i)} = 2$.

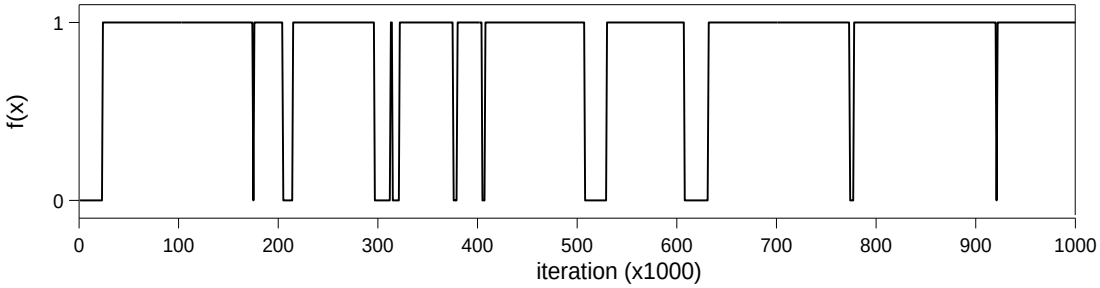


Figure 5-3: Plot of the values of $f(x)$ for every 1,000 iterations for a chain of length 1,000,000 simulated using the Gibbs sampler.

5.3 Sampling the posterior distribution of Example 4 using the Gibbs sampler

In order to sample from the target distribution $\pi(x)$ defined by equation (5.2.1), we ran the Gibbs sampler for 1,000,000 iterations, with a burn-in of 1,000 iterations. The values of X at iterations 10,000, 20,000 and 30,000 are given in Figure 5-2. From this figure we can see that very obvious modes exist, connected to the appearance and disappearance of certain pixels or features. We shall investigate the behaviour of the 9×9 pixels at the lower left-hand corner, which we shall denote by region B. Let

$$f(x) = I\left\{\frac{\sum_{i \in B} I(x^{(i)} = 2)}{81} \geq 0.2\right\}.$$

Thus $f(x) = 0$ when region B has predominantly $X^{(i)}$ values equal to 1 but $f(x) = 1$ when at least 20% of the pixels in B have $X^{(i)} = 2$. The values of $f(x)$ for every 1,000 iterations for the chain of length 1,000,000 are shown in Figure 5-3. From this figure we can see that the Gibbs sampler is mixing slowly with respect to the feature in region B.

5.4 Mode jumping using modelling

5.4.1 General methodology

We shall apply our mode jumping approach to the image restoration problem, using the MJM method with implicit modelling. Note that in this problem, we define "modes" as images that have the local maximum posterior probabilities, i.e., a state x is a mode if $\pi(x)$ will decrease if we change the value $x^{(i)}$ of any pixel. To apply this method, we go through the three stages:

1. Initial exploration, where we try to find all possible locations of the modes.
2. Clustering, where we cluster the mode locations and find a single mode location to represent each cluster.
3. Sampling, where we use the mode locations from the clustering stage to jump between the modes.

Initial exploration

To find all possible mode locations we use simulated annealing, where we work with a target distribution proportional to $\pi(x)^{1/T(k)}$ on the k^{th} iteration. Here $T(k)$ is our chosen temperature function. Note that if $f_X(x) = b\,\pi(x)^{1/T(k)}$, for the appropriate value of b to make this a probability distribution, then

$$\begin{aligned} f_X(x) &= b\,\pi(x)^{1/T(k)} \\ &= b\left\{\pi_{X^{(i)}|X^{(-i)}}(x^{(i)}|x^{(-i)})\,\pi_{X^{(-i)}}(x^{(-i)})\right\}^{1/T(k)}. \end{aligned}$$

Therefore,

$$\begin{aligned} f_{X^{(i)}|X^{(-i)}}(x^{(i)}|x^{(-i)}) &= \frac{f_X(x)}{f_{X^{(i)}}(x^{(i)})} \\ &= \frac{b\left\{\pi_{X^{(i)}|X^{(-i)}}(x^{(i)}|x^{(-i)})\right\}^{1/T(k)}\left\{\pi_{X^{(-i)}}(x^{(-i)})\right\}^{1/T(k)}}{f_{X^{(-i)}}(x^{(-i)})} \\ &= b'\left\{\pi_{X^{(i)}|X^{(-i)}}(x^{(i)}|x^{(-i)})\right\}^{1/T(k)}, \end{aligned}$$

where b' depends on $x^{(-i)}$ but is constant with respect to $x^{(i)}$. This means that to simulate from a target distribution which is proportional to $\pi(x)^{1/T(k)}$ we can simulate from the conditional distributions $\pi_{X^{(i)}|X^{(-i)}}(x^{(i)}|x^{(-i)})^{1/T(k)}$ instead, which is equivalent to running the Gibbs sampler with conditional distributions raised to the power $1/T(k)$ and re-normalised. For our case, this involves simulating, on the k^{th}

cycle and at the i^{th} pixel, from the conditional distribution

$$P_{T(k)}(x^{(i)}|y, x^{(-i)}) = b' \left\{ P_{X^{(i)}|Y,X^{(-i)}}(x^{(i)}|y, x^{(-i)}) \right\}^{\frac{1}{T(k)}},$$

where b' is the appropriate value to make this a probability distribution.

After running the Gibbs sampler with simulated annealing to a conclusion, we shall apply a "hill-climbing" process to find the nearest mode. In this hill-climbing process, we sweep over the image from $X^{(1)}$ to $X^{(p^2)}$, and if changing the value of $X^{(i)}$ to the other value in the set $\{1, 2\}$ will increase the posterior probability of the whole image X, then $X^{(i)}$ is changed to this value; this process is repeated until convergence. We repeat the procedure of simulated annealing and hill-climbing l times using a specified set of starting values to provide us with mode locations X_j, $j = 1, \ldots, l$.

Clustering

We define the "distance" between two mode locations X_i and X_j, a_{ij}, to be the number of pixels at which values differ, i.e.,

$$a_{ij} = \sum_{k=1}^{p^2} I\{x_i^{(k)} \neq x_j^{(k)}\}. \tag{5.4.1}$$

Using this definition of distance, we cluster the l possible mode locations that we found in the initial exploration stage using the hierarchical clustering algorithm as described in section 4.1.1 with stopping criterion ξ. At the end of this algorithm the clusters will be a distance of at least ξ from each other.

The clustering algorithm produces a number, m, of distinct mode locations. Our aim is to discover all sufficiently distinct modes of the posterior distribution. A binary image can be thought of as a collection of objects or "features" set against a plain background, for example, the images in Figure 5-2 contain features in the bottom left and top right corners. Given the form of $\pi(x)$, we would expect there to be "modal" versions of such features that may be present or absent in a modal image X. If there are q distinct features, 2^q images maybe created from them. It is possible that the initial search produces a subset of these 2^q images because images with some particular combinations of the features are not generated during the initial exploration stage. As long as each of the q features can be identified by comparing pairs within the m modes found, all 2^q possible combinations of the features can be recovered by applying the "cross-over" operation (Franconi & Jennison (1993)) to the m mode locations. We shall illustrate an example of a cross-over operation later on in section 5.4.2.

Sampling

We define the "nearest mode" function, $h(x) \in \{1, \ldots, m'\}$, so that it will choose the nearest mode j which has the minimum distance between X and η_j, using the definition of distance given by equation (5.4.1). The sampling stage then proceeds as described in section 3.3.4, where the mode jumping step follows the MJM method with implicit modelling, therefore using a cycle of Gibbs sampling as the final step in generating a proposal. Note that here we shall use z to denote the proposal state instead of y; we define the probability of moving from a mode location η_j to a proposal state z via one cycle of the Gibbs sampler to be

$$\begin{aligned} q_j(z) &= \pi_{X^{(1)}|X^{(-1)}}(z^{(1)}|\eta_j^{(2)}, \ldots, \eta_j^{(p^2)}) \times \pi_{X^{(2)}|X^{(-2)}}(z^{(2)}|z^{(1)}, \eta_j^{(3)}, \ldots, \eta_j^{(p^2)}) \times \\ &\quad \ldots \times \pi_{X^{(p^2)}|X^{(-p^2)}}(z^{(p^2)}|z^{(1)}, \ldots, z^{(p^2-1)}) \\ &= \prod_{i=1}^{p^2} \pi_{X^{(i)}|X^{(-i)}}(z^{(i)}|z^{(1)}, \ldots, z^{(i-1)}, \eta_j^{(i+1)}, \ldots, \eta_j^{(p^2)}). \end{aligned}$$

5.4.2 Application to Example 4

We apply our MJM method to sample from the target distribution $\pi(x)$ defined by equation (5.2.1), and compare the performance of our mode jumping approach with that of the Gibbs sampler seen in section 5.3.

Implementation

1. Initial exploration

   We carried out a run of length 100 using the Gibbs sampler with simulated annealing, using the logarithmic temperature function, $T(k) = 3/\ln(1+k)$, and applied the hill-climbing process to the end image. We repeated the procedure of simulated annealing and hill-climbing 100 times, where 50 runs started with $X = \{1\}$ and the other 50 with $X = \{2\}$. These runs produced 100 mode locations, including 61 distinct ones. A few examples of the mode locations that we found are shown in Figure 5-4.

2. Clustering

   For the clustering algorithm, we chose $\xi = 7$. Consequently, the 61 distinct mode locations were reduced to $m = 4$. The corresponding mode locations η_i, $i = 1, \ldots, 4$, are shown in Figure 5-5. We then applied a cross-over operation to the η_i. Let $\eta_i^{\{L\}}$ denote the left-hand half of the image η_i and $\eta_i^{\{R\}}$ the right-hand half, and denote the whole image by $\eta_i = (\eta_i^{\{L\}}, \eta_i^{\{R\}})$. The four modal images provide two versions of $\eta_i^{\{L\}}$ and two versions of $\eta_i^{\{R\}}$. The cross-over exercise generates 4 images as combinations $\eta_{kl} = (\eta_k^{\{L\}}, \eta_l^{\{R\}})$ for k, $l = 1, \ldots, 4$. After rejecting mode locations which are just repetitions, we ended up with $m' = 2^2$

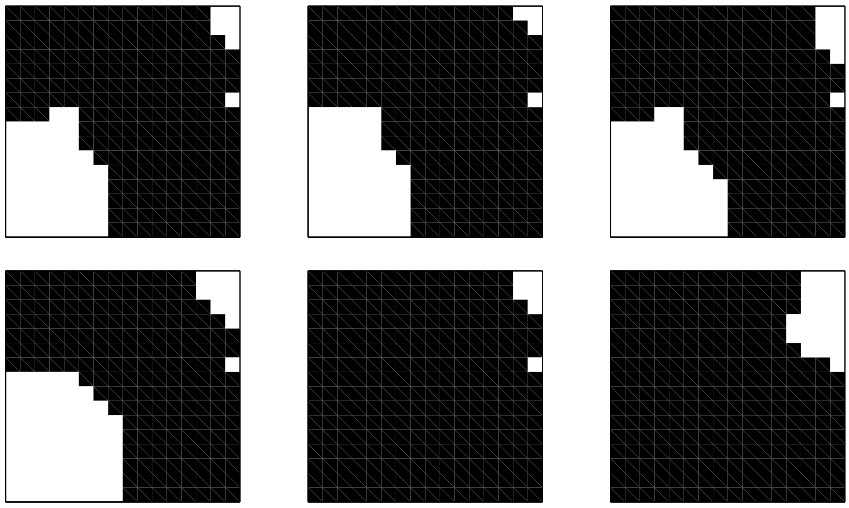
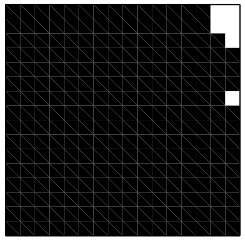


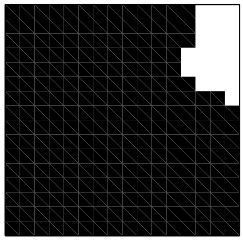
Figure 5-4: A few examples of the mode locations found in the initial exploration stage, where pixel i is black for $X^{(i)} = 1$ and white for $X^{(i)} = 2$.

(a) η_1

(b) η_2



(c) η_3



(d) η_4

Figure 5-5: The mode locations after clustering, where pixel i is black for $\eta^{(i)} = 1$ and white for $\eta^{(i)} = 2$.

mode locations, which are the same as before. However, if we had not found one of the four η_i in the clustering stage, the cross-over operation would have provided this additional mode location.

3. Sampling

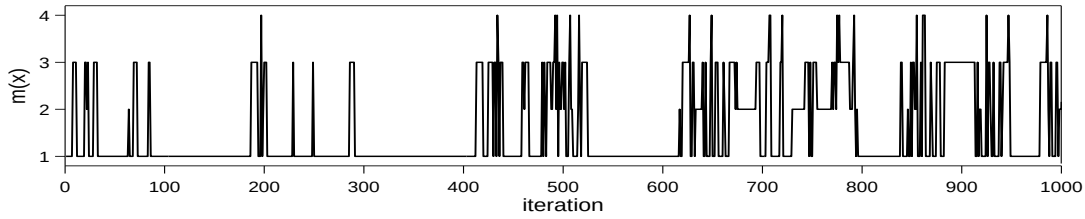
   In the sampling stage, we mix local movements using the Gibbs sampler and mode jumping steps: at each iteration we choose to do Gibbs sampling with probability θ and to do a mode jumping step with probability $1-\theta$. The mode jumping step proceeds as given in section 5.4.1, where we choose to jump to a different mode with equal probability, i.e., $p_{ij} = 1/(m'-1)$ for $j \in \{1, \ldots, m'\}$, $j \neq i$.

Results for Example 4

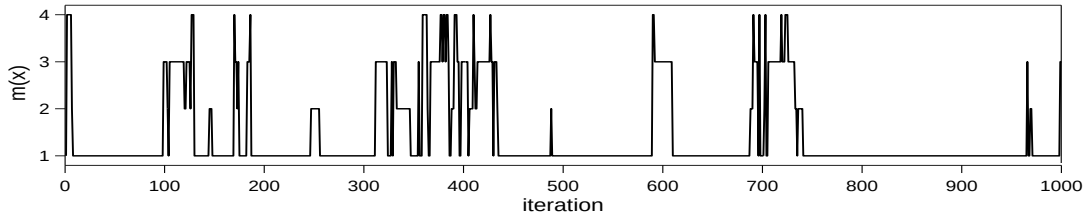
We compare sampling from the target distribution $\pi(x)$ using the MJM method and using the Gibbs sampler. Note that in this chapter, when we compare any sampling methods we shall use the comparison criteria as described in section 4.2.2. In doing this, we define the nearest mode function $m(x)$ so that it will choose the nearest mode j to x among $\eta_1, \ldots, \eta_4$ as shown in Figure 5-5 with distance as defined by equation (5.4.1).

We simulated chains of length 1,000,000 using the MJM method and the Gibbs sampler (after a burn-in of 1,000 iterations). For the Gibbs sampler, we start with a random starting point, while for the mode jumping method we start with a random mode location. For our mode jumping method, we used $\theta = 0.0$, 0.25, 0.5, and 0.75. The values of $m(x)$ for the first 1,000 iterations after burn-in of the various simulated chains are shown in Figure 5-6. From this figure, we can see that the chain simulated using the Gibbs sampler moves nicely between modes 1 and 2 and moves well between modes 3 and 4, but it is very poor at moving between the pair (1,2) and the pair (3,4). We can also see this poor mixing of the Gibbs sampler from the estimate of the transition matrix for the movement of the chain between the four modes. Estimates for these two methods based on these 1,000,000 iterations are given by

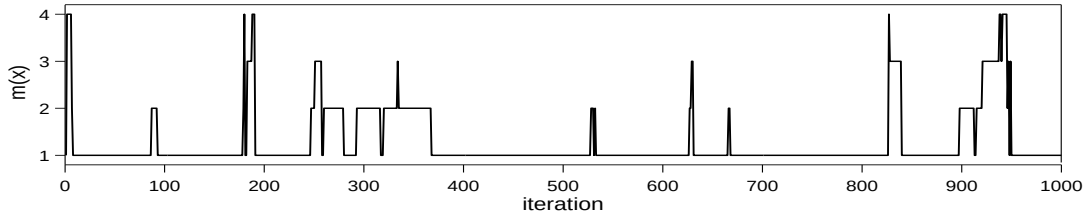
$$\widehat{P}_{gs} = \begin{pmatrix} 0.9831 & 0.0166 & 0.0003 & 0.0 \\ 0.0963 & 0.9023 & 0.0 & 0.0003 \\ 0.0021 & 0.0 & 0.9812 & 0.0166 \\ 0.0003 & 0.0024 & 0.0928 & 0.9045 \end{pmatrix}$$



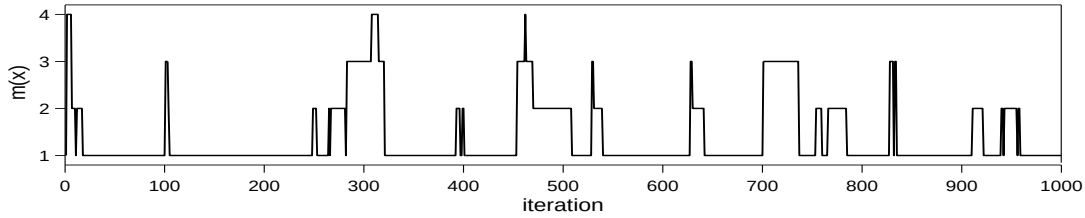
(a)(i) the MJM method with $\theta = 0.0$



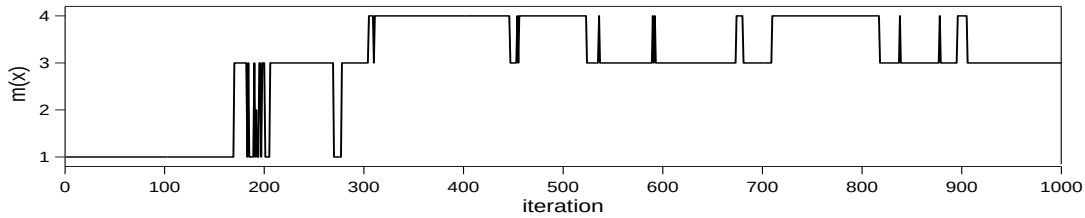
(a)(ii) the MJM method with $\theta = 0.25$



(a)(iii) the MJM method with $\theta = 0.5$



(a)(iv) the MJM method with $\theta = 0.75$



(b) the Gibbs sampler

Figure 5-6: Plots of the values of $m(x)$ for the first 1,000 iterations, where the chains are simulated using (a) the MJM method with various values of θ; and (b) the Gibbs sampler.

	Method				
	MJM				Gibbs
	$\theta = 0.0$	$\theta = 0.25$	$\theta = 0.5$	$\theta = 0.75$	sampler
Mode jumping rate (%)	10.67	8.51	6.65	4.79	2.89
Block jumping rate (%)	7.60	5.58	3.77	1.94	0.05
Total computation ($\times 10^6$)					
a) initial exploration					
i) Gibbs cycles	0.01	0.01	0.01	0.01	0
b) sampling					
i) Gibbs cycles	2	1.75	1.5	1.25	1
ii) evaluations of $\pi(z)/\pi(x)$	1	0.75	0.5	0.25	0
Rate of convergence, λ^*	0.832	0.855	0.874	0.918	0.997
Estimated IAC for $f^{(i)} = I\{m(x) = i\}$ calculated using $\widehat{P}$					
1) $\hat{\tau}(f^{(1)})$	10.04	12.13	14.73	20.46	318.25
2) $\hat{\tau}(f^{(2)})$	9.52	11.21	12.64	14.06	31.03
3) $\hat{\tau}(f^{(3)})$	5.11	6.96	10.52	20.94	635.35
4) $\hat{\tau}(f^{(4)})$	1.89	2.76	4.11	7.57	117.16

Table 5.1: Overall results for a chain of run length 1,000,000, simulated using the MJM method and the Gibbs sampler.

for the Gibbs sampler, and

$$\widehat{P}_m = \begin{pmatrix} 0.9605 & 0.0141 & 0.0205 & 0.0049 \\ 0.0847 & 0.8451 & 0.0507 & 0.0195 \\ 0.1454 & 0.0604 & 0.7555 & 0.0387 \\ 0.1997 & 0.1352 & 0.2191 & 0.4460 \end{pmatrix}$$

for the MJM method with $\theta = 0.25$. We can see that there are larger values off the diagonal in $\widehat{P}_m$, including significant probabilities for moves from (1,2) to (3,4) and back. The overall results for both methods are summarized in Table 5.1; we shall discuss these results one by one.

From Table 5.1, we can see that our MJM method yields higher mode jumping rates than the Gibbs sampler. This can also be observed from Figure 5-6. In fact, for the case where $\theta = 0$ our mode jumping method will successfully jump between modes at least three times as frequently as the Gibbs sampler. The trace for the Gibbs sampler in Figure 5-6 and the values off the diagonal in $\widehat{P}_{gs}$ indicate particular difficulty in moving between modes 1 and 2 and modes 3 and 4. Denote modes 1 and 2 together as block A, and modes 3 and 4 as block B. We define the "block jumping rate" as the rate of jumping between blocks A and B, corresponding to changing the block of pixels in the bottom left-hand corner of the image. From Table 5.1, we can see that

the block jumping rates for the MJM method are much higher than that for the Gibbs sampler. In fact, the block jumping rate for our mode jumping method with $\theta = 0$ is at least 150 times higher than that for the Gibbs sampler. This shows that the MJM method is mixing very much better than the Gibbs sampler.

We also need to compare the amount of computation used by each method. In this case, there are two significant computations: 1) the p^2 steps making up one cycle of the Gibbs sampler; and 2) the evaluation of $\pi(z)/\pi(x)$ in the acceptance probability $\alpha(x, z)$. The Gibbs sampler uses one cycle of the Gibbs sampler update in each iteration. In the case of the MJM method, in each iteration we choose to do Gibbs sampling with probability θ and to do a mode jumping step with probability $1 - \theta$. For each mode jumping step we use two cycles of the Gibbs sampler and one evaluation of $\pi(z)/\pi(x)$. This means that each iteration the MJM method uses on average $(2-\theta)$ Gibbs sampler updates and $(1-\theta)$ evaluations of $\pi(z)/\pi(x)$. If we look at the total computation used by both methods as given in Table 5.1, then sampling using the mode jumping using mode locations method costs more per iteration than sampling using the Gibbs sampler.

We can gauge the relative cost of the two types of computation by observing that one evaluation of $\pi(z)/\pi(x)$ is equivalent to at most one cycle of the Gibbs sampler update. This is because we can rewrite $\pi(z)/\pi(x)$ as the product of p^2 terms,

$$\frac{\pi(z)}{\pi(x)} = \frac{\pi(z(1))}{\pi(x)} \times \frac{\pi(z(2))}{\pi(z(1))} \times \ldots \frac{\pi(z(n))}{\pi(z(p^2-1))}, \tag{5.4.2}$$

where $z(i)$ is the vector x with components $1, 2, \ldots, i$ updated, which means $z(p^2) = z$. In one cycle of the Gibbs sampler update, we update each pixel i, $i = 1, \ldots, p^2$, by calculating the probability of a state $x^{(i)}$, $P_{X^{(i)}|Y,X^{(-i)}}(x^{(i)}|y, x^{(-i)})$, which is equal to

$$\frac{\pi(x|X^{(i)} = x^{(i)})}{\pi(x|X^{(i)} = x^{(i)})) + \pi(x|X^{(i)} \neq x^{(i)}))}.$$

This is equivalent to calculating a term $\pi(z(i))/\pi(z(i-1))$ in expression (5.4.2) for $\pi(z)/\pi(x)$. Therefore the calculations in one cycle of the Gibbs sampler update are sufficient to evaluate $\pi(z)/\pi(x)$. Since each mode jumping step uses two cycles of the Gibbs sampler update and one evaluation of $\pi(z)/\pi(x)$, this means that one mode jumping step costs 3 times as much as one cycle of the Gibbs sampler update. Then computation for the MJM method with parameter θ is equivalent to $(3-2\theta)$ times that of one Gibbs sampler update. The higher mode jumping rate and much higher block jumping rate of the MJM method suggest it is still much more effective than the Gibbs sampler; we shall explore this further as we consider other aspects of efficiency.

We have treated $\widehat{P}$ as an estimate of P, the transition matrix of the process $\{m(X_t)\}$

which we approximate to be Markov. Hence we estimate the rate of convergence, λ^*, for this process by the second largest in absolute value eigenvalue of $\widehat{P}$. From Table 5.1, we can see values of λ^* for the MJM method are lower than that for the Gibbs sampler, indicating faster convergence to the correct distribution over the four modes for the MJM method. The fact that the value of λ^* for the Gibbs sampler is very close to one means that overall the chain simulated using the Gibbs sampler is converging very slowly.

The values of the IAC, $\tau(f)$, for $f^{(i)}(x) = I\{m(x) = i\}$, $i = 1, \ldots, 4$, which are estimated using the corresponding $\widehat{P}$, are also given in Table 5.1. We can see that the values of $\hat{\tau}(f^{(i)})$ for the MJM method are lower than those for the Gibbs sampler. In particular, the value of $\hat{\tau}(f^{(3)})$ for the MJM method with $\theta = 0$ is less than one hundredth of the value for the Gibbs sampler, i.e., our method estimates $\mathbb{P}(m(x) = 3)$ with the same estimation accuracy as the Gibbs sampler using just one hundredth of the run length. This means that the MJM method has better estimation performance than the Gibbs sampler. However, we have to take into account the computational cost of each method. If we combine this with the fact that the MJM method with $\theta = 0$ costs at most 3 times as much as the Gibbs sampler, then overall this method is still at least 33 times more efficient than the Gibbs sampler.

In general, the MJM method performs more efficiently than the Gibbs sampler when applied to this particular example. Our mode jumping approach does have extra set up costs, which needs to be considered in efficiency comparisons. In the initial exploration stage, our method uses one Gibbs sampler cycle for each iteration of the simulated annealing. However, with the extent of initial exploration used in this example the total amount of computation used in the initial exploration stage is only a small fraction of the total number used in the sampling stage; for example, for the case where $\theta = 0$, the computation used in the initial exploration stage is only 0.3% of the computation used in the sampling stage. This means that the initial exploration is relatively cheap and accounting for it makes little difference to the efficiency comparisons noted above. On the other hand, we do have to run the initial exploration long enough to ensure that the probability of missing a mode is low. In our initial exploration stage, we do multiple short runs, where half of those start with $X = \{1\}$ and the other half with $X = \{2\}$. Then in the clustering stage, the cross-over operation protects against missing one, or possibly two, of the four modes. The settings in our initial exploration stage and the cross-over operation will ensure that the probability of missing a mode is low. We could also increase the number of multiple short runs in the initial exploration stage substantially to reduce the probability of missing a mode, with little effect on the total computation.

5.5 Mode jumping using differences — the first version (MJD1)

5.5.1 General methodology

We can also apply the usual MJD method, which we shall call method MJD1, to the image restoration problem. To do this, we shall go through the three stages: 1) initial exploration; 2) clustering; and 3) sampling. The initial exploration and clustering stages are the same as the ones in the MJM method, as described in section 5.4.1, while the sampling stage follows as described in section 3.3.4. Note that for the sampling stage we shall again use z to denote the proposal state instead of y, and correspondingly use z' instead of y'. For the sampling stage we define the difference between modes j and k, $d_{j,k} = \eta_k - \eta_j$. We take the operator $\oplus$ to be a restricted addition so that for $z' = x \oplus d_{h(x),j}$, we have

$$z'^{(i)} = \max\left(\min\left[x^{(i)} + d^{(i)}_{h(x),j},\, 2\right],\, 1\right)$$

for $i = 1, \ldots, p^2$, as $x^{(i)} \in \{1, 2\}$. We set the random perturbation process to be an update using one cycle of the Gibbs sampler, so that in this case $q(z', z)$ is the probability of moving from state z' to the proposal state z via the cycle of Gibbs sampling.

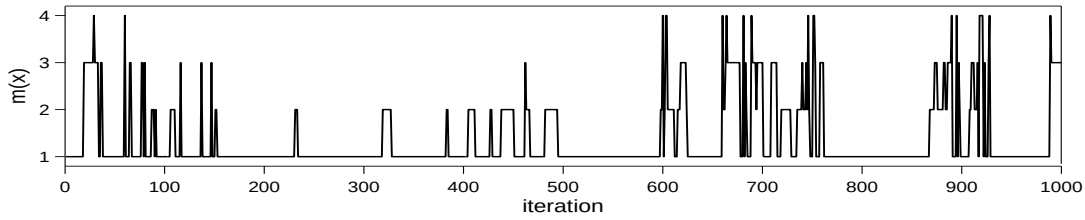
5.5.2 Application to Example 4

We simulated a chain of length 1,000,000 using the method MJD1, starting from a random mode location. For this mode jumping method, we again used $\theta = 0.0$, 0.25, 0.5, and 0.75. The values of $m(x)$ for the first 1,000 iterations of the various simulated chains are shown in Figure 5-7. In this figure, we can see that the chains simulated using method MJD1 are mixing fairly well. The estimate from these 1,000,000 iterations of the transition matrix for the movement of the chain between the four modes for the case where $\theta = 0.25$ is given by

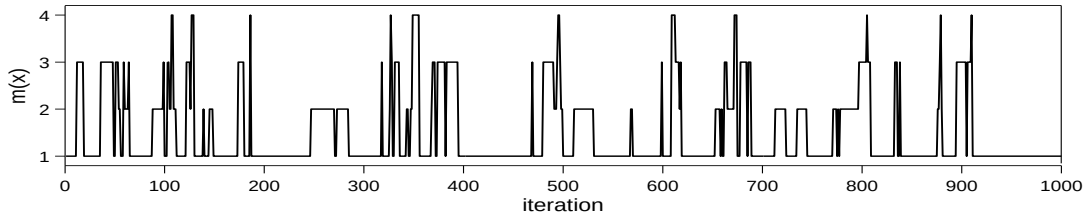
$$\widehat{P}_{d1} = \begin{pmatrix} 0.9550 & 0.0206 & 0.0199 & 0.0044 \\ 0.1236 & 0.8201 & 0.0410 & 0.0153 \\ 0.1358 & 0.0498 & 0.7779 & 0.0364 \\ 0.1789 & 0.1017 & 0.2079 & 0.5115 \end{pmatrix},$$

which is similar to the transition matrix when the chain is simulated using the MJM method (refer to $\widehat{P}_m$ in section 5.4.2). The overall results for this method are summarized in Table 5.2.

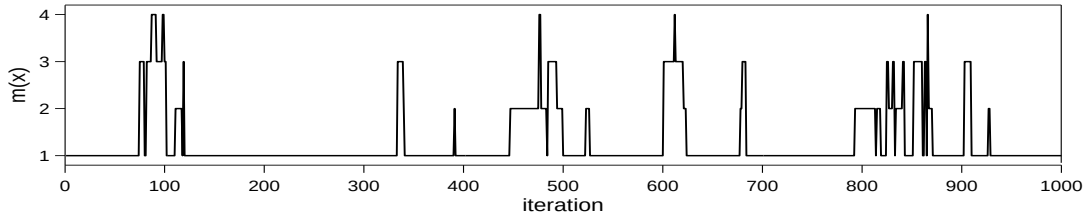
From this table, we can see that the performance of this method is similar to that



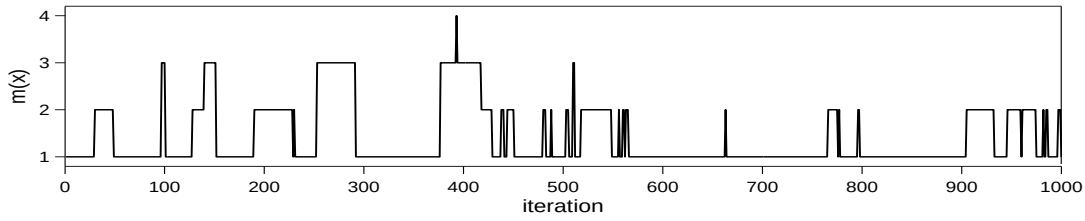
(a) $\theta = 0.0$



(b) $\theta = 0.25$



(c) $\theta = 0.50$



(d) $\theta = 0.75$

Figure 5-7: Plots of the values of $m(x)$ for the first 1,000 iterations, where the chains are simulated using the method MJD1 with various values of θ.

	Method				
	MJD1				Gibbs
	$\theta = 0.0$	$\theta = 0.25$	$\theta = 0.50$	$\theta = 0.75$	sampler
Mode jumping rate (%)	10.48	8.94	6.83	4.85	2.89
Block jumping rate (%)	6.38	5.06	3.36	1.71	0.05
Total computation ($\times 10^6$)					
a) initial exploration					
i) Gibbs cycles	0.01	0.01	0.01	0.01	0
b) sampling					
i) Gibbs cycles	2	1.75	1.5	1.25	1
ii) evaluations of $\pi(z)/\pi(x)$	1	0.75	0.5	0.25	0
Rate of convergence, λ^*	0.798	0.824	0.864	0.924	0.997
Estimated IAC for $f^{(i)} = I\{m(x) = i\}$ calculated using $\widehat{P}$					
1) $\hat{\tau}(f^{(1)})$	8.54	10.29	13.37	20.06	318.25
2) $\hat{\tau}(f^{(2)})$	7.95	9.02	10.78	12.89	31.03
3) $\hat{\tau}(f^{(3)})$	5.62	7.63	11.84	22.94	635.35
4) $\hat{\tau}(f^{(4)})$	2.22	3.23	4.74	8.62	117.16

Table 5.2: Overall results for a chain of run length 1,000,000, simulated using the method MJD1 and the Gibbs sampler.

of the MJM method as given in Table 5.1. This means that the method MJD1 also performs better than the Gibbs sampler when applied to this particular example. On the other hand, like the MJM method, method MJD1 also uses two full cycles of the Gibbs sampler in each mode jumping step, which is very costly. One way we can reduce the computational cost of the mode jumping step in the method MJD1 is to concentrate just on the pixels which are changed when we move from the current x to $z' = x \oplus d_{h(x),j}$, i.e., the pixels i where $d^{(i)}_{h(x),j} \neq 0$. Then instead of doing a full cycle of Gibbs sampler on z' to get a proposal z, we can do a restricted cycle of the Gibbs sampler on z' where we only update pixels i where $d^{(i)}_{h(x),j} \neq 0$. Doing a restricted cycle of the Gibbs sampler instead of a full cycle in the mode jumping step will greatly reduce the computational cost of our method, especially when the number of pixels i where $d^{(i)}_{h(x),j} \neq 0$ is small compared to the whole image. To do a restricted cycle of Gibbs sampler would then require us to identify the area where $d^{(i)}_{j,k} \neq 0$ for all j, $k = 1, \ldots, m'$, $k \neq j$. However, there is an alternative approach to mode jumping using differences which uses these areas directly in jumping between modes. We shall discuss this alternative approach in the next section.

5.6 Mode jumping using differences - the second version (MJD2)

5.6.1 General methodology

In the previous section we presented a mode jumping method using the differences $d_{j,k} = \eta_k - \eta_j$. It is these differences that we need to make moves, rather than the modes η_i themselves. The "differences" are liable to comprise sets of distinct features, as mentioned in section 5.4.1 in the discussion of "cross-over" within clustering. If we can identify such features in their simplest form, we have the building blocks for creating all modal images by adding or removing these features. This is the basis of our second version of mode jumping using differences, which we shall call method MJD2.

To apply method MJD2 to the image restoration problem, we go through the three stages: 1) initial exploration; 2) clustering; and 3) sampling. The initial exploration stage is the same as the one in the method MJD1, as described in section 5.5.1.

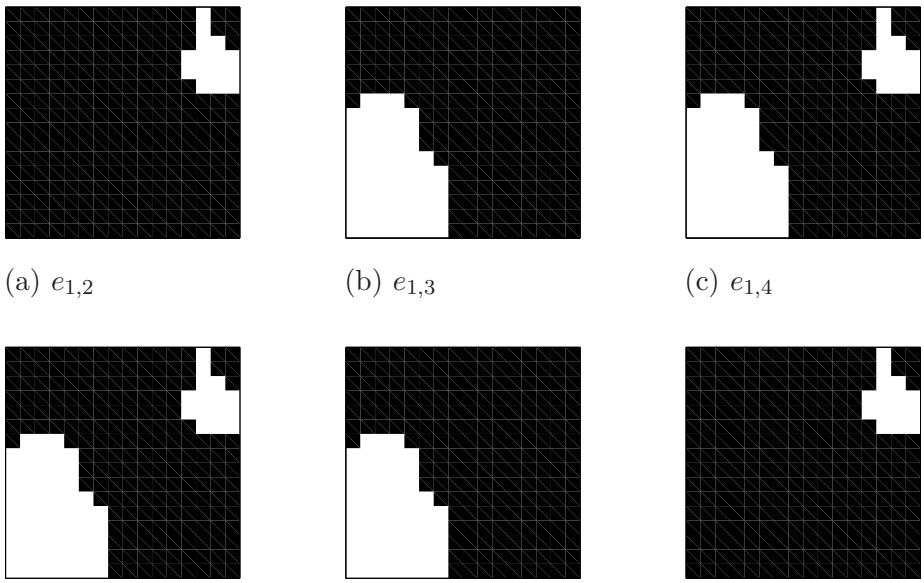
Clustering

The clustering stage starts by following the clustering procedure of section 5.5.1 to produce the mode locations η_j, $j = 1, \ldots, m$. However, we continue to work with the set of η_js as follows. First, for each pair of j and k, where $j, k \in \{1, \ldots, m\}$, $j < k$, we define $e_{j,k}$ with elements

$$e_{j,k}^{(i)} = I\{\eta_j^{(i)} \neq \eta_k^{(i)}\}, \quad i = 1, \ldots, p^2.$$

Therefore the set of pixels i for which $e_{j,k} = 1$ represents the region of differences between the two modes η_j and η_k. We want to group these regions of differences into sets of connecting pixels. We shall illustrate this by using an example.

Consider the η_j, $j = 1, \ldots, 4$, given in Figure 5-5. The corresponding $e_{j,k}$ that we obtained from these η_j are shown in Figure 5-8. If any $e_{j,k}$ contains a single connected set of pixels taking the value 1, we generate a vector $W = e_{j,k}$, where $W^{(i)} = 1$ if i is in the connected set of pixels and $W^{(i)} = 0$ otherwise; therefore W represents a feature that may or may not be present in different modes. Examples of this case are $e_{1,2}$ and $e_{1,3}$. If any $e_{j,k}$ contains more than one connected set of pixels taking the value 1, we separate these sets and produce one W for each set. Examples of this case are $e_{1,4}$ and $e_{2,3}$. Therefore, for $e_{1,4}$, we construct a W with elements equal to 1 for pixels in the bottom left white area and 0 otherwise, and we create another W with elements equal to 1 for pixels in the upper right white area. We collect all the Ws generated from all the $e_{j,k}$s and cluster them using the same hierarchical clustering algorithm as described in section 5.4.1, using the definition of distance given by equation (5.4.1) and



(a) $e_{1,2}$ (b) $e_{1,3}$ (c) $e_{1,4}$

(d) $e_{2,3}$ (e) $e_{2,4}$ (f) $e_{3,4}$

Figure 5-8: The region of differences $e_{j,k}$, where pixels i with $e_{j,k}^{(i)} = 0$ are coloured black and those with $e_{j,k}^{(i)} = 1$ are coloured white.

a stopping criterion parameter ξ_2. We choose the W with the largest number of pixels i where $W^{(i)} = 1$ to represent each cluster. This clustering process will then remove any duplicates and group together any Ws which are similar to each other. After going through this process, we are left with W_k, $k = 1, \ldots, m_2$. These W_k then represent individual features which are present or absent in the mode locations η_j. We shall now use these W_k in the sampling stage.

Sampling

We shall use the vectors W_k to make proposals for jumps between modes. We do this by defining the functions

$$g(x, W_k) = I\left\{\frac{\sum_i I\left\{(W_k^{(i)} = 1)\bigcap(x^{(i)} = 2)\right\}}{\sum_i I(W_k^{(i)} = 1)} \geq 0.5\right\}.$$

Thus $g(x, W_k) = 1$ when x takes the value 2 on at least 50% of the pixels i where $W_k^{(i)} = 1$, and $g(x, W_k) = 0$ otherwise. Then for one mode jumping step in this method, we:

1. Choose a vector W_k, $k \in \{1, \ldots, m_2\}$, with equal probability for all m_2 choices.

2. Determine the value of $g(x, W_k)$ for the current x.

(a) If $g(x, W_k) = 0$, set $z'^{(i)} = \min(x^{(i)} + W_k^{(i)}, 2)$ for $i = 1, \ldots, p^2$.

(b) If $g(x, W_k) = 1$, set $z'^{(i)} = \max(x^{(i)} - W_k^{(i)}, 1)$ for $i = 1, \ldots, p^2$.

This has the effect of making all values of x equal to a common value on the pixels where $W_k^{(i)} = 1$ and leaving the rest of x unchanged. Then we do one restricted cycle of the Gibbs sampler on z', where we only update pixels i where $W_k^{(i)} = 1$, to get a proposal z. Let $q(z', z)$ denote the probability of moving from state z' to the proposal state z via the restricted Gibbs sampler.

3. Consider the reverse jump. We determine the value of $g(z, W_k)$.

(a) If $g(z, W_k) = 0$, set $x'^{(i)} = \min(x^{(i)} + W_k^{(i)}, 2)$ for $i = 1, \ldots, p^2$.

(b) If $g(z, W_k) = 1$, set $x'^{(i)} = \max(x^{(i)} - W_k^{(i)}, 1)$ for $i = 1, \ldots, p^2$.

We calculate $q(x', x)$, the probability of x' changing to the current image x via the restricted Gibbs sampler. We then accept the proposal z with probability

$$\alpha(x, z) = \min\left\{1, \frac{\pi(z)\, q(x', x)}{\pi(x)\, q(z', z)}\right\}.$$

This mode jumping step satisfies detailed balance following the same argument as in the original MJD method, as discussed in section 3.3.4. The computation in implementing a mode jumping step by this method is greatly reduced as we concentrate on the pixels i where $W_k^{(i)} = 1$ and only doing a restricted cycle of the Gibbs sampler. The simplification is greatest when the number of pixels i where $W_k^{(i)} = 1$ is small compared to the whole image.

The vectors W_k represent features that can either be absent or present in a mode. If we consider all possible combinations of the features represented by the W_k, $k = 1, \ldots, m_2$, we shall end up with 2^{m_2} modal images. All that is necessary to achieve this is that each feature is generated as a W_k from a difference in two modes, η_j and η_l say, found in the initial exploration stage, and all that is necessary for this to happen is that the feature should be present in one mode and absent from another. Thus, we have achieved the effect of performing an automated version of the "cross-over" process described in section 5.4.1 but by a different mechanism. This helps ensure that the probability of our method missing a mode is low. The method MJD2 then is able to jump between these 2^{m_2} modes automatically through its own implicit version of the cross-over operation.

(a) W_1 (b) W_2

Figure 5-9: Vectors W_k, $k = 1, 2$, where pixel i is black for $W_k^{(i)} = 0$ and white for $W_k^{(i)} = 1$.

5.6.2 Application to Example 4

Implementation

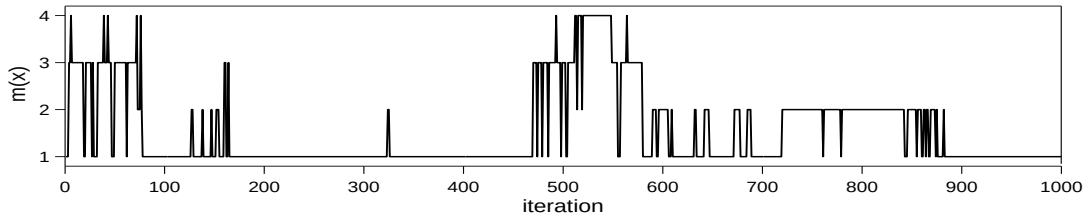
We apply the method MJD2 to sample from the target distribution $\pi(x)$ defined by equation (5.2.1). Implementation of the initial exploration and clustering stages follows as in section 5.4.2. For the mode jumping step in the sampling stage, we use the method MJD2 described in the previous section. To find the vectors W_k, we chose $\xi_2 = 3$. Consequently, we obtained $m_2 = 2$; the vectors W_k, $k = 1, 2$, are shown in Figure 5-9.

Results for Example 4

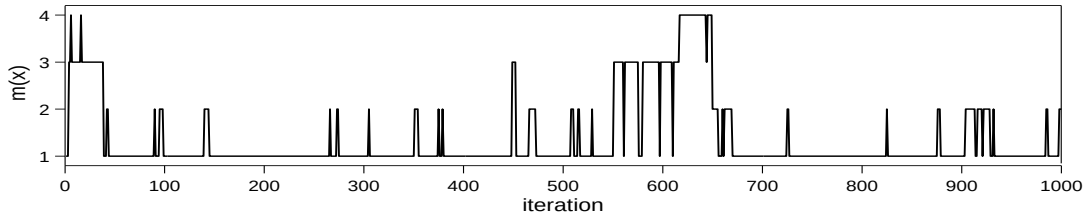
We simulated a chain of length 1,000,000 using the method MJD2, starting from a random mode location. We choose $\theta = 0.1$, 0.25, 0.5, and 0.75. Here we chose $\theta = 0.1$ rather than $\theta = 0$ because if we were to use $\theta = 0$ then the pixels not in one of the W_ks would never get updated. The values of $m(x)$ for the first 1,000 iterations of the various simulated chains are shown in Figure 5-10. In this figure, we can see that the chains simulated using method MJD2 are mixing fairly well. However, the trace for method MJD2 in Figure 5-10 shows a different pattern of mode jumps to the trace for the MJM method in Figure 5-6 and the trace for method MJD1 in Figure 5-7. This is also evident when we compare estimates from the 1,000,000 iterations of the transition matrices for movements between modes for these three methods. The estimated transition matrix for method MJD2 with $\theta = 0.25$ is

$$\widehat{P}_{d2} = \begin{pmatrix} 0.9563 & 0.0339 & 0.0098 & 0.0 \\ 0.1963 & 0.7945 & 0.0 & 0.0092 \\ 0.0686 & 0.0 & 0.8970 & 0.0343 \\ 0.0 & 0.0650 & 0.2110 & 0.7239 \end{pmatrix}.$$

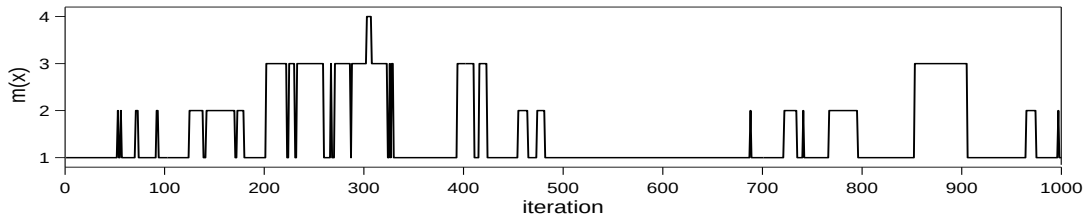
Comparing this with $\widehat{P}_m$ in section 5.4.2 for the MJM method and $\widehat{P}_{d1}$ in section 5.5.2 for method MJD1, we see there are smaller values off the diagonal in $\widehat{P}_{d2}$. In particular, there are zero probabilities for moves from modes 1 to 4, modes 2 to 3, modes 3 to



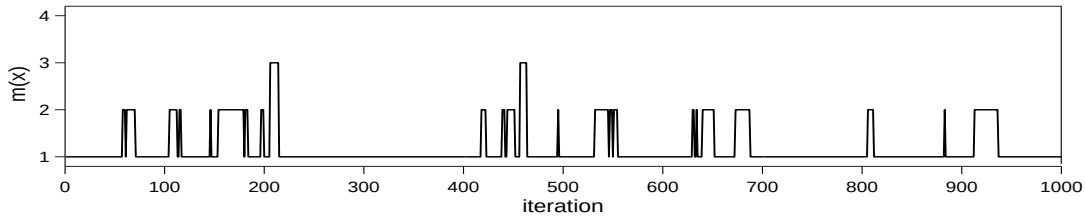
(a) $\theta = 0.10$



(b) $\theta = 0.25$



(c) $\theta = 0.50$



(d) $\theta = 0.75$

Figure 5-10: Plots of the values of $m(x)$ for the first 1,000 iterations, where the chains are simulated using the method MJD2 with various values of θ.

	Method				
	MJD2				Gibbs
	$\theta = 0.10$	$\theta = 0.25$	$\theta = 0.50$	$\theta = 0.75$	sampler
Mode jumping rate (%)	8.21	7.49	5.92	4.46	2.89
Block jumping rate (%)	2.00	1.69	1.15	0.61	0.05
Total computation ($\times 10^6$)					
a) initial exploration					
i) Gibbs cycles	0.01	0.01	0.01	0.01	0
b) sampling					
i) Gibbs cycles	0.361	0.468	0.645	0.823	1
ii) evaluations of $\pi(z)/\pi(x)$	0.9	0.75	0.5	0.25	0
Rate of convergence, λ^*	0.907	0.922	0.945	0.972	0.997
Estimated IAC for $f^{(1)} = I\{m(x) = i\}$ calculated using $\widehat{P}$					
1) $\hat{\tau}(f^{(1)})$	12.48	14.55	20.99	36.89	318.25
2) $\hat{\tau}(f^{(2)})$	7.04	7.80	9.57	12.85	31.03
3) $\hat{\tau}(f^{(3)})$	17.90	21.56	32.24	60.59	635.35
4) $\hat{\tau}(f^{(4)})$	7.17	7.94	11.20	18.44	117.16

Table 5.3: Overall results for a chain of run length 1,000,000, simulated using the method MJD2 and the Gibbs sampler.

2 and modes 4 to 1. These are to be expected since jumps are not possible directly for modes 1 and 4 and modes 2 and 3, as these jumps involve changing *both* features. From $\widehat{P}_{d2}$ we also see that moves associated with W_2, i.e., moves from modes 1 to 3, modes 3 to 1, modes 2 to 4 and modes 4 to 2, have low probabilities.

The overall results for the method MJD2 are summarized in Table 5.3. The overall mode jumping rates for method MJD2 as given in Table 5.3 are similar to those for the MJM method in Table 5.1 and those for method MJD1 in Table 5.2. However, the block jumping rates for method MJD2 are low compared with the other two methods (less than one third). The blocks are defined as modes 1 and 2 and modes 3 and 4, therefore this is consistent with the low values identified in $\widehat{P}_{d2}$. This means that proposals to add or remove the large feature in the lower left-hand corner are being rejected with high probability.

In the case of the method MJD2 we increase computational efficiency by just focusing on the regions where the differences are present, i.e., doing a restricted cycle of the Gibbs sampler. In each iteration we choose to do Gibbs sampling with probability θ and to do a mode jumping step with probability $1-\theta$, where for each mode jumping step we are using two restricted cycles of the Gibbs sampler update and one evaluation of $\pi(z)/\pi(x)$. This means that method MJD2 uses θ Gibbs sampler updates, $2(1-\theta)$

restricted Gibbs sampler updates and $1-\theta$ evaluations of $\pi(z)/\pi(x)$ per iteration. In this example, the regions of difference only involve on average 37 pixels which is only 14.5% of the whole image, therefore the restricted cycle of the Gibbs sampler costs only 14.5% of the usual Gibbs sampler. This means that this method uses $\theta + 2(1-\theta) \times 0.145 = 0.29 + 0.71\theta$ Gibbs sampler updates and $1-\theta$ evaluations of $\pi(z)/\pi(x)$ per iteration. If we look at the total computational cost used by method MJD2 as given in Table 5.3 and the cost used by the MJM method and method MJD1 as given in Table 5.1 and Table 5.2 respectively, then sampling using method MJD2 costs far less than sampling using the other two methods.

From Table 5.3 and the previous Table 5.1 and Table 5.2, we can see that the movement between modes for method MJD2 has slower convergence rates, i.e., larger values of λ^*, than the other two methods. The values of the IAC, $\tau(f)$, for $f^{(i)}(x) = I\{m(x) = i\}$, $i = 1, \ldots, m$, which is estimated using the corresponding $\widehat{P}$, are also given in these three tables. We can see that method MJD2 produces values of $\hat{\tau}(f^{(i)})$ which are higher than those produced by the other two methods. From these results we can see that although method MJD2 costs less per iteration than the other two methods, it performs worse. For example, if we compare the value of $\hat{\tau}(f^{(3)})$ for the three methods with $\theta = 0.25$ combined with the computation required, then the MJM method and method MJD1 are around 1.5 times more efficient than method MJD2. But method MJD2 still does perform better than the Gibbs sampler. To improve the performance of method MJD2, we shall propose a slightly different version of this method in the next section, which achieves most of the savings in computation while attaining low values for λ^* and the $\hat{\tau}(f^{(i)})$.

5.7 Mode jumping using differences - the third version (MJD3)

5.7.1 General methodology

One reason why method MJD2 could be having problems is because after adding or subtracting W_k from the current x, we do a restricted Gibbs sampler on the changed values, i.e., we only update the pixels that have been changed. We have to consider the pixels with $W_k^{(i)} = 0$ but which are neighbours of pixels with $W_k^{(i)} = 1$. Suppose x had mostly 2s in the $W_k^{(i)} = 1$ region, therefore the proposal z has mostly 1s in this region. We expect the next layer of neighbouring pixels, where $W_k^{(i)} = 0$, will have some 2s present due to the 2s that x had in pixels with $W_k^{(i)} = 1$. Now that z has 1s for those pixels, the 2s in the layer of neighbours have low conditional probability. This means that z is not a very likely sample under π, resulting in a low value of $\pi(z)$ and consequently a low acceptance probability for z.

One way to overcome this problem is to turn z into a more likely sample under π by expanding the restricted Gibbs sampler to include the pixels with $W_k^{(i)} = 0$ which are neighbours of pixels with $W_k^{(i)} = 1$. For example, we can do the restricted Gibbs sampler on an area which is bigger than the set $\{i : W_k^{(i)} = 1\}$ by one layer of surrounding, connecting pixels. We can generate such an area by using the methods of mathematical morphology (Matheron (1975) and Serra (1982)). Each vector W_k is then treated as a $p \times p$ image and is processed using a "dilation" operation (see for example chapter 5 of Glasbey & Horgan (1995)) with a square of size 3×3 as the structuring element (the reference pixel is at the center) to generate a corresponding vector $R_k = \{R_k^{(i)}\}$, where $R_k^{(i)} \in \{0, 1\}$. We can think of the dilation operation as putting a copy of the structuring element at every pixel i where $W_k^{(i)} = 1$, with the reference pixel exactly on i. In doing this we shall generate an R_k for which the set $\{i : R_k^{(i)} = 1\}$ is bigger than $\{i : W_k^{(i)} = 1\}$ by one layer of surrounding, connecting pixels. Doing the restricted Gibbs sampler on the sets of pixels i where $R_k^{(i)} = 1$ instead of the set where $W_k^{(i)} = 1$ will then produce a more likely sample under $\pi(x)$.

Then to apply a modified version of method MJD2, which we shall call method MJD3, to the image restoration problem, we shall go through the three stages: 1) initial exploration; 2) clustering; and 3) sampling. The initial exploration, clustering and sampling stages are the same as the ones in the method MJD2, as described in section 5.6.1. However, in a mode jumping step in the sampling stage, instead of doing the restricted Gibbs sampler on the set of pixels i where $W_k^{(i)} = 1$, we do it on the set of pixels i where $R_k^{(i)} = 1$.

5.7.2 Application to Example 4

Implementation

We apply the method MJD3 to sample from the target distribution $\pi(x)$ defined by equation (5.2.1). The vectors R_k, $k = 1, 2$, used in mode jumping steps are shown in Figure 5-11.

Results for Example 4

We simulated a chain of length 1,000,000 using the method MJD3, starting with a random mode location. We again choose $\theta = 0.1$, 0.25, 0.5, and 0.75. The values of $m(x)$ for the first 1,000 iterations of the various simulated chains are shown in Figure 5-12. In this figure, we can see that the chains simulated using this method are mixing better than the chains simulated using method MJD2 as given in Figure 5-10. This is also evident when we compare the estimate of the transition matrix for the movement

(a) R_1 (b) R_2

Figure 5-11: Vectors R_k, $k = 1, 2$, where pixel i is black for $R_k^{(i)} = 0$ and white for $R_k^{(i)} = 1$.

of the MJD3 chain with $\theta = 0.25$ between the four modes, which is given by

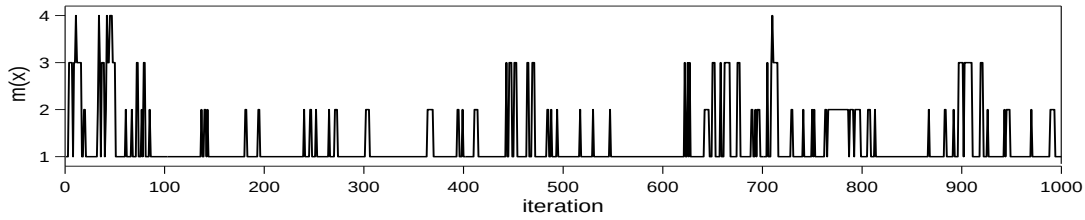
$$\widehat{P}_{d3} = \begin{pmatrix} 0.9039 & 0.0547 & 0.0414 & 0.0 \\ 0.3226 & 0.6365 & 0.0 & 0.0409 \\ 0.2903 & 0.0 & 0.6557 & 0.0540 \\ 0.0 & 0.2909 & 0.3190 & 0.3899 \end{pmatrix},$$

with $\widehat{P}_{d2}$ in section 5.6.2 for method MJD2. We can see that there are larger values off the diagonal in $\widehat{P}_{d3}$.

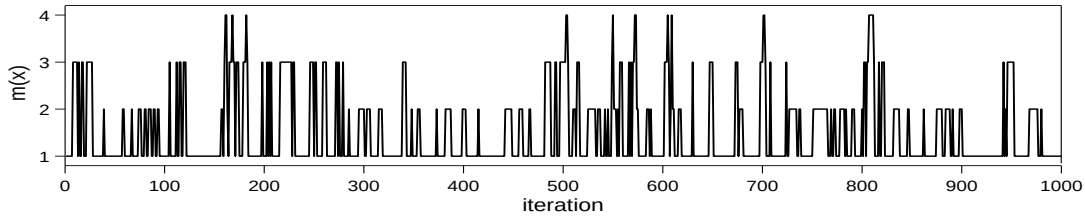
The overall results for the method MJD3 are summarized in Table 5.4. If we compare the mode jumping rates for this method with those for method MJD2 as given in Table 5.3, we see that the mode jumping rates for this method are more than twice as high. Furthermore, the block jumping rates for MJD3 are more than 4 times as high as those for MJD2.

In the method MJD3 the area where $R_k^{(i)} = 1$ involves on average 54.5 pixels, therefore on average the restricted cycle of the Gibbs sampler now costs 21.3% of the usual Gibbs sampler, which is an increase of 6.8% when compared to its cost in method MJD2. This means that method MJD3 uses $\theta + 2(1-\theta) \times 0.213 = 0.426 + 0.574\theta$ Gibbs sampler updates and $1 - \theta$ evaluations of $\pi(z)/\pi(x)$. If we look at the total computational cost used by method MJD3 as given in Table 5.4 and the cost used by method MJD2 as given in the previous Table 5.3, then sampling using method MJD3 costs slightly more than method MJD2.

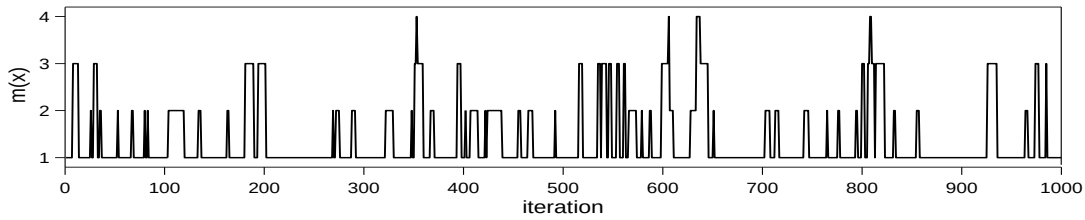
From Table 5.4 and the previous Table 5.3, we can see that the movement between modes for method MJD3 has much higher convergence rates, i.e., smaller values of λ^* than method MJD2. The estimated values of the IAC, $\tau(f)$, for $f^{(i)} = I\{m(x) = i\}$, $i = 1, \ldots, m$, which is estimated using the corresponding $\widehat{P}$, are also given in these two tables. From these tables, we can see that for the case where $\theta = 0.1$ method MJD3



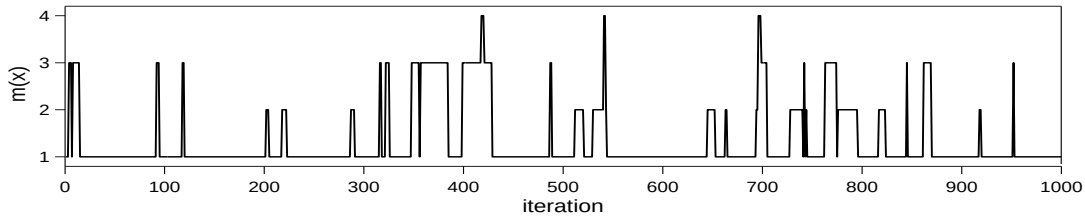
(a) $\theta = 0.10$



(b) $\theta = 0.25$



(c) $\theta = 0.50$



(d) $\theta = 0.75$

Figure 5-12: Plots of the values of $m(x)$ for the first 1,000 iterations, where the chains are simulated using the method MJD3 with various values of θ.

	Method				
	MJD3				Gibbs
	$\theta = 0.10$	$\theta = 0.25$	$\theta = 0.50$	$\theta = 0.75$	sampler
Mode jumping rate (%)	19.11	16.58	11.96	7.45	2.89
Block jumping rate (%)	8.50	7.25	4.81	2.47	0.05
Total computation ($\times 10^5$)					
a) initial exploration					
i) Gibbs cycles	0.01	0.01	0.01	0.01	0
b) sampling					
i) Gibbs cycles	0.483	0.570	0.713	0.857	1
ii) evaluations of $\pi(z)/\pi(x)$	0.9	0.75	0.5	0.25	0
Rate of convergence, λ^*	0.605	0.668	0.779	0.889	0.997
Estimated IAC for $f^{(i)} = I\{m(x) = i\}$ calculated using $\widehat{P}$					
1) $\hat{\tau}(f^{(1)})$	3.69	4.43	6.61	12.12	318.25
2) $\tau(f^{(2)})$	3.42	4.02	5.63	8.75	31.03
3) $\tau(f^{(3)})$	3.63	4.56	7.27	15.08	635.35
4) $\tau(f^{(4)})$	2.01	2.52	3.91	7.64	117.16

Table 5.4: Overall results for a chain of run length 1,000,000, simulated using the method MJD3 and the Gibbs sampler.

produces estimates of the same accuracy as method MJD2 or better using just half of the run length. Given that the extra computational cost is slight, MJD3 is superior overall.

Comparing the results for the method MJD3 with those for the MJM method as given in Table 5.1 and those for method MJD1 as given in Table 5.2, we can see that the mode jumping rates for method MJD3 are higher (nearly twice as high), while the block jumping rates are approximately the same for the three methods. Taking one evaluation of $\pi(z)/\pi(x)$ as equal to one cycle of the Gibbs sampler update, the total computation used for sampling by method MJD3 with $\theta = 0.1$ is only 46% of that used by the other two methods with $\theta = 0.0$. From the values of $\hat{\tau}(f^{(i)})$, we can see that method MJD3 produces estimates of the same accuracy as the other two methods using similar or shorter run lengths but the computational cost per iteration is lower. This means that method MJD3 is more efficient than the MJM method and method MJD1 when applied to this example.

The method MJD3 performs better than the MJM method and the method MJD1 because it concentrates on changing the pixels in just one area of the image when making a proposal while leaving the rest unchanged. It breaks up mode jumps into basic parts — changing one feature, representing a set of connected pixels, at a time.

It also leaves alone pixels away from the feature being changed, taking advantage of the fact that their values are a typical sample under π.

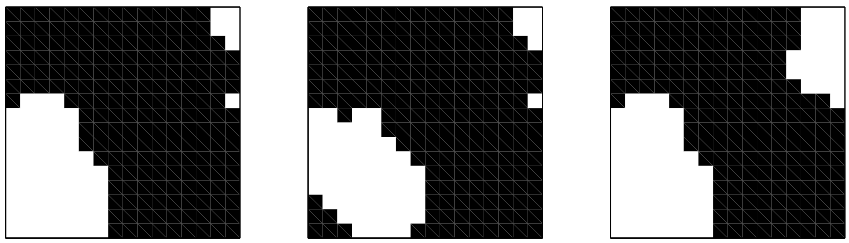
5.7.3 Data analysis based on the MCMC results

Since the method MJD3 performs the best among all the methods that we have proposed and discussed, we shall analyze the archeological data as given in Example 4 based on the results of this method. Using 1,000,000 iterations simulated from the method MJD3 with $\theta = 0.25$, we calculated estimates of the probabilities that the true image is closest to each of the modal images $\eta_1, \ldots, \eta_4$ given in Figure 5-5, which are 0.75, 0.13, 0.10 and 0.02 respectively. Therefore in the Bayesian analysis we can say that there is a posterior probability of 0.88 of activity in the bottom left-hand corner, and there is a posterior probability close to one of activity in some parts of the top right-hand corner but the posterior probability that this extends over the majority of the larger region shown in Figure 5-5 (b) and (d) is only 0.15.

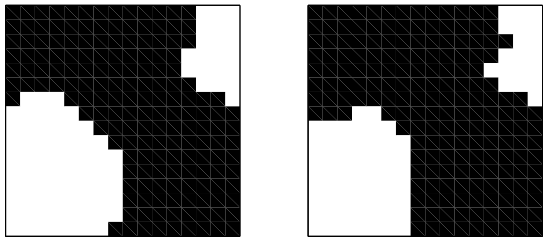
However, we can go further and use the 1,000,000 iterations to estimate the probabilities associated with the features W_1 and W_2 given in Figure 5-9 being predominantly absent in the true image X, i.e., the probability that X takes the value 2 on less than 20% of the pixels listed in W_1 and W_2. The estimated probabilities that we obtained for W_1 and W_2 being mostly absent are 0.64 and 0.12 respectively. This means that there is a fairly high probability that there is no activity in the area W_1, and there is some probability that there is no activity in the area W_2.

5.8 Applying the method MJD3 to Example 4 with alternative parameter values

The values of β and σ^2 that were used in Example 4 as described in section 5.2, i.e., $\beta = 0.78$ and $\sigma^2 = 0.5$, were chosen for illustration purposes as they lead to problems for the Gibbs sampler, and this presented a challenging "mixing" problem for our mode jumping methods to overcome. We shall now consider the actual values that were used by Gray (1994), $\beta = 0.738$ and $\sigma^2 = 0.434$. We shall use these parameter values and apply method MJD3 to sample from the target distribution $\pi(x)$ defined by equation (5.2.1). We shall see that with these values for β and σ^2 the posterior distribution has more modes but method MJD3 is particularly good at sampling from all the modes. We shall also compare the results for method MJD3 with those of the Gibbs sampler.



(a) η_1 (b) η_2 (c) η_3



(d) η_4 (e) η_5

Figure 5-13: The mode locations after clustering, where pixel i is black if $\eta_j^{(i)} = 1$ and white if $\eta_j^{(i)} = 2$.

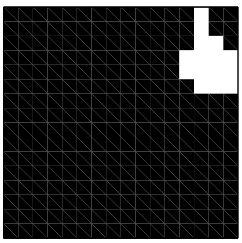
5.8.1 Implementation and results

Initial exploration

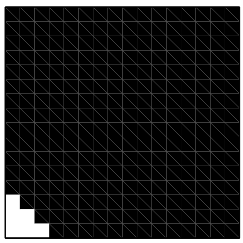
We carried out a run of length 500 using the Gibbs sampler with simulated annealing, using the temperature function $T(k) = 3/\ln(1+k)$, and applied the hill-climbing process to the end image. We repeated the procedure of simulated annealing and hill-climbing 100 times, where 50 runs started with $X = \{1\}$ and the other 50 with $X = \{2\}$. These runs produced 100 mode locations, including 59 distinct ones.

Clustering

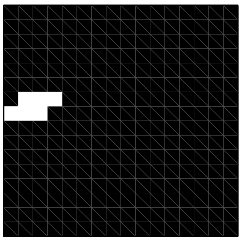
For the clustering algorithm, we chose $\xi = 5$. Consequently, the 59 distinct mode locations were reduced to $m = 5$. The corresponding mode locations η_i, $i = 1, \ldots, 5$, are shown in Figure 5-13. If we were to apply the cross-over operation to these mode locations, wherein each mode location is separated horizontally into two equal sections, we would obtain four different objects for the section on the left, and three different objects for the right, which would result in 12 different modes. However, method MJD3 achieves an equivalent effect by alternative means.



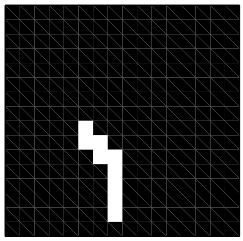
(a) W_1



(b) W_2



(c) W_3



(d) W_4

Figure 5-14: Vectors W_k, $k = 1, \ldots, 4$, where pixel i is black if $W_k^{(i)} = 0$ and white if $W_k^{(i)} = 1$.

Obtaining the vectors W_k

We chose $\xi_2 = 5$ and consequently obtained $m_2 = 4$; the W_k, $k = 1, \ldots, 4$, are shown in Figure 5-14. Note that if we were to use these W_k in a cross-over operation where we consider all possible combinations of the modes with the vectors W_k, $k = 1, \ldots, 4$, where each W_k can either be absent of present in a mode, then we shall end up with $2^4 = 16$ modes (refer to Figure 5-15). Here we obtain $2^3 = 8$ possibilities for the section on the left, i.e., more than the 4 cases seen in the original $\eta_1, \ldots, \eta_5$ shown in Figure 5-13, but we only obtain $2^1 = 2$ objects on the right — fewer than the 3 in the original $\eta_1, \ldots, \eta_5$. This is because the right-hand feature in the original η_5 is missing, as it is close (within 5 pixels) of the feature in the original η_4. The reason both of these survived clustering with $\xi = 5$ is that the left-hand feature in the original η_5 differs from other left-hand features and the clustering process considers *total* differences in the whole image, whereas clustering of W_ks is based on the number of pixel differences in each feature.

Nevertheless, the modes that we obtain by using W_k in a cross-over operation are more than the modes that we can find by applying the cross-over operations to the mode locations η_j themselves, where we can only obtain 12. Applying the cross-over operation to the mode locations directly only produced 12 modes because some combinations did not appear in the initial exploration. However, we consider *all* possible combinations if

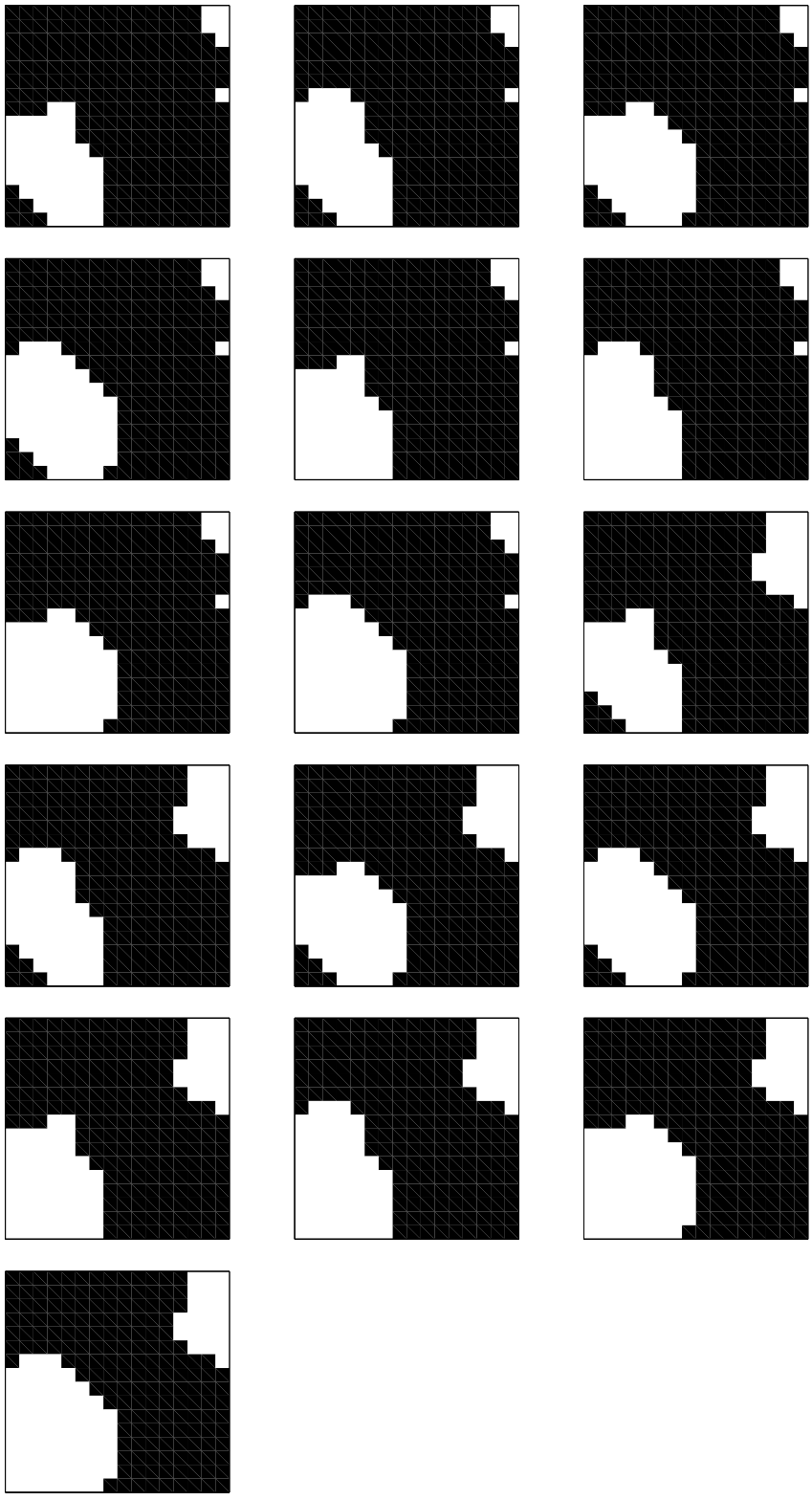


Figure 5-15: The mode locations η_j obtained from the 16 possible combinations of the W_ks. Pixel i is coloured black if $\eta_j^{(i)} = 1$ and white if $\eta_j^{(i)} = 2$.

we apply the cross-over operation using the differences. In this respect, using differences instead of mode locations directly seems to have an advantage. The method MJD3 also ensures that we can jump between these 16 modes, which will also ensure that the probability of our method missing a mode is low.

Sampling

To sample from the target distribution $\pi(x)$, we combine local movements using the Gibbs sampler and mode jumping steps: at each iteration we choose to do Gibbs sampling with probability θ and to do a mode jumping step with probability $1 - \theta$. The mode jumping step then proceeds as described in section 5.7.1.

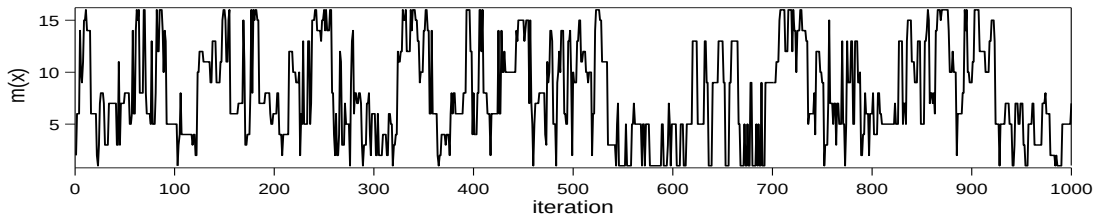
To assess method MJD3 for this example, we define the nearest mode function $m(x)$ so that it will choose the nearest mode j to x among $\eta_1, \ldots, \eta_{16}$ as shown in Figure 5-15 with distance as defined by equation (5.4.1). We simulated a chain of length 1,000,000 using method MJD3 and the Gibbs sampler (after a burn-in of 1,000 iterations). For our mode jumping method, we choose $\theta = 0.1$, 0.25, 0.5, and 0.75. The values of $m(x)$ for the first 1,000 iterations of the various simulated chains are shown in Figure 5-16. From this figure, we can see that the chain simulated using the Gibbs sampler makes less frequent moves between the set of modes 1 to 8 and the set of modes 9 to 16.

The overall results for both methods are summarized in Table 5.5. For this example we denote the set of modes 1 to 8 as block A and the set of modes 9 to 16 as block B, and we define the "block jumping rate" as the rate of jumping between blocks A and B. From Table 5.5, we can see that although the block jumping rates for method MJD3 are higher than for the Gibbs sampler, both methods have has similar mode jumping rates. This means that the Gibbs sampler still mixes quite well.

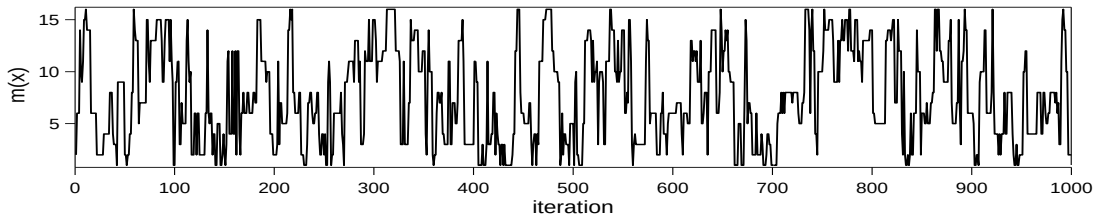
From Table 5.5, we can see that the values of λ^* for method MJD3 are lower than for the Gibbs sampler, therefore convergence to the correct distribution over the 16 modes is faster for method MJD3. On the other hand, the values of $\hat{\tau}(f^{(i)})$ shown in Table 5.5 show that method MJD3 produces estimates of the same accuracy as the Gibbs sampler using similar run length. Therefore, in this example the method MJD3 performs at least as well as the Gibbs sampler. However, one clear advantage of the method MJD3 over the Gibbs sampler is that using this method we can be confident we know about all the different modes in the distribution that we are trying to sample from, and that this method is able to sample from these modes.

5.8.2 Data analysis based on the MCMC results

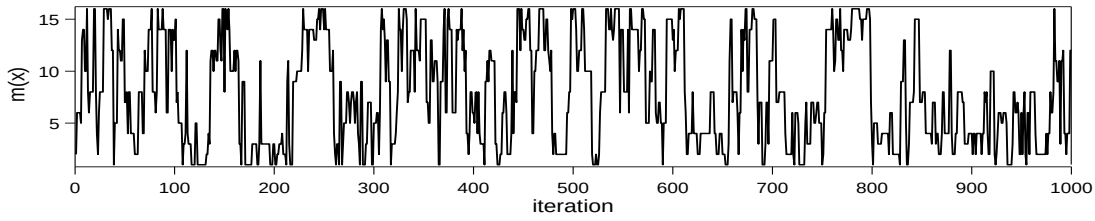
We shall analyze the archeological data as given in Example 4 based on the results when sampling using the parameter values $\beta = 0.738$ and $\sigma^2 = 0.434$. From the modal



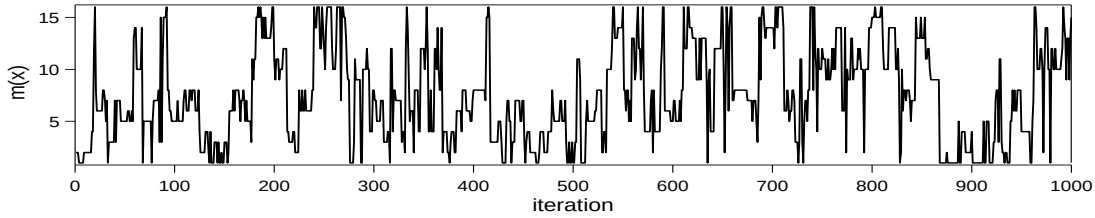
(a)(i) the method MJD3 with $\theta = 0.1$



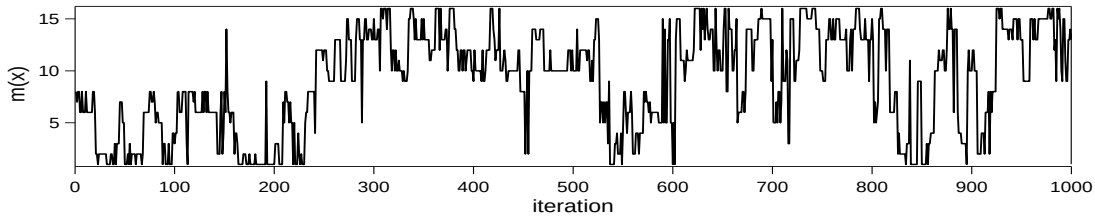
(a)(ii) the method MJD3 with $\theta = 0.25$



(a)(iii) the method MJD3 with $\theta = 0.5$



(a)(iv) the method MJD3 with $\theta = 0.75$



(b) the Gibbs sampler

Figure 5-16: Plots of the values of $m(x)$ for the first 1,000 iterations, where the chains are simulated using (a) the method MJD3 with various values of θ; and (b) the Gibbs sampler.

	Method				
	MJD3				Gibbs
	$\theta = 0.1$	$\theta = 0.25$	$\theta = 0.5$	$\theta = 0.75$	
Mode jumping rate (%)	45.91	46.05	46.05	45.99	46.16
Block jumping rate (%)	11.85	10.82	8.91	7.12	5.27
Total computation ($\times 10^6$)					
a) initial exploration					
i) Gibbs cycles	0.05	0.05	0.05	0.05	0
b) sampling					
i) Gibbs cycles	0.269	0.391	0.594	0.797	1
ii) evaluations of $\pi(z)/\pi(x)$	0.9	0.75	0.5	0.25	0
Rate of convergence, λ^*	0.797	0.777	0.816	0.853	0.891
Estimated IAC for $f^{(i)} = I\{m(x) = i\}$ calculated using $\widehat{P}$					
1) $\hat{\tau}(f^{(1)})$	4.07	3.90	3.88	3.95	4.14
2) $\hat{\tau}(f^{(2)})$	3.95	3.89	3.90	3.98	4.28
3) $\hat{\tau}(f^{(3)})$	3.66	3.53	3.45	3.48	3.59
4) $\hat{\tau}(f^{(4)})$	4.02	3.84	3.82	3.86	4.13
5) $\hat{\tau}(f^{(5)})$	4.31	4.18	4.05	4.12	4.33
6) $\hat{\tau}(f^{(6)})$	4.24	4.13	4.05	4.19	4.52
7) $\hat{\tau}(f^{(7)})$	3.94	3.75	3.67	3.58	3.74
8) $\hat{\tau}(f^{(8)})$	4.29	4.13	4.01	4.10	4.35
9) $\hat{\tau}(f^{(9)})$	3.65	3.65	3.73	3.88	4.31
10) $\hat{\tau}(f^{(10)})$	3.64	3.62	3.72	4.03	4.52
11) $\hat{\tau}(f^{(11)})$	3.35	3.30	3.32	3.46	3.67
12) $\hat{\tau}(f^{(12)})$	3.60	3.63	3.68	3.90	4.27
13) $\hat{\tau}(f^{(13)})$	3.93	3.84	3.88	4.13	4.53
14) $\hat{\tau}(f^{(14)})$	3.90	3.91	4.04	4.24	4.78
15) $\hat{\tau}(f^{(15)})$	3.57	3.48	3.46	3.67	3.96
16) $\hat{\tau}(f^{(16)})$	3.94	3.86	3.97	4.20	4.62

Table 5.5: Overall results for a chain of run length 1,000,000, simulated using the method MJD3 and the Gibbs sampler.

images $\eta_1, \ldots, \eta_{16}$ given in Figure 5-15, we can see that there is a very high probability that there is at least some activity in the bottom left-hand corner and top right-hand corner. Therefore in this case we are more interested in the probabilities associated with the features $W_1, \ldots, W_4$ given in Figure 5-14 being predominantly absent in the true image X, i.e., the probability that X takes the value 2 on less than 20% of the pixels listed in W_k, $k = 1, \ldots, 4$. The estimated probabilities that we obtained for $W_1, \ldots, W_4$ being absent are 0.34, 0.30, 0.25 and 0.13 respectively. This means that there is some probability that there is no activity in the areas $W_1, \ldots, W_4$.

Note that the conclusions that we came to here are different from the ones we obtained when sampling using the parameter values $\beta = 0.78$ and $\sigma^2 = 0.5$ as described in section 5.7.3. In particular, in section 5.7.3 there was some probability of no activity at all in the lower left corner. This is not surprising since the posterior distribution of the true image X given the data Y is influenced by the values of the parameters β and σ^2. A more sophisticated Bayes analysis would include a prior distribution for β and σ^2. This approach is not without difficulty as the normalising constants for $\pi(x)$ under different values of β have to be computed.

5.9 Summary and discussion

We have shown in an image analysis problem how the mode jumping approach can be applied to target distributions with discrete variables. We can make our methods work even better through some adaption to the problem, and we found the method MJD3 to perform particularly well. Although our mode jumping approach does have extra set up cost, the initial exploration and processing of the modes found in this search are relatively cheap compared to the cost of running the Markov chain to sample from the target distribution.

One clear advantage of our mode jumping approach over the basic Gibbs sampler is that we can be confident we know about the different modes in the distribution that we are trying to sample from, and that our method is able to sample from these modes. Although the Tjelmeland & Hegstad (2001) mode jumping method was not designed for the discrete case, a modified version can be constructed to give mode jumping in the discrete case (see Sharp (2003), chapter 5). In this mode jumping method, the mode jumping step consists of a "big step", for example changing the values of a randomly chosen set of pixels, optimization via hill-climbing and sampling via a cycle of the Gibbs sampler; in the reverse "big step" we reverse the values of the same randomly chosen set of pixels. However, without knowledge from an initial search for possible modes this mode jumping step is unlikely to make a really good big step — just as in chapter 4. Furthermore, the reverse jump will not necessarily go back to the original mode.

In our mode jumping approach, the methods which involve differences seem to work particularly well in this problem. This can be attributed to the fact that "modes" in this problem involve certain features that appear or disappear, and it is sufficient for a method to change only certain pixels while keeping the rest fixed. An example of another method which also does this is the Swenden & Wang (1987) method, which updates stochastically generated blocks using the Gibbs sampler. However, the MJD method has a clear advantage over their method because the initial exploration and clustering stages have already identified the area that needs block updating in order to move between the modes. The Swenden & Wang (1987) method, on the other hand, generates these blocks randomly without paying special attention to specific features that one would like to see appear and disappear during sampling from the posterior distribution.

Chapter 6

Application III: A Mixture of Discrete and Continuous Variables

In this chapter, we shall consider an application in Bayesian linear regression which leads to a sampling problem where the target distribution has a mixture of discrete and continuous variables. We shall apply our mode jumping approach to this problem and assess its performance.

6.1 Outliers in the Bayesian linear regression model

In the simple Bayesian linear regression model, the observations $\boldsymbol{Y} = (Y^{(1)}, Y^{(2)}, \ldots, Y^{(n)})^{\mathrm{T}}$ are generated by

$$\boldsymbol{Y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where n is the sample size, X is an $n \times p$ matrix of nonrandom variables, $\boldsymbol{\beta}$ is a $p \times 1$ vector of unknown parameters and $\boldsymbol{\varepsilon}$ is an $n \times 1$ vector of independent normal errors with mean zero and variance σ^2. It is common practice to use an improper prior distribution for the location and scale parameters with density $P_{\beta,\sigma^2}(\beta, \sigma^2) = \sigma^{-2}$.

Bayesian methods for fitting linear regression models in the presence of outliers assume a model for the generation of *all* the data set, including the possible outliers. The more frequently analyzed model is the *normal scale contamination model* (Box & Tiao (1968)), where the error distribution is

$$\varepsilon^{(i)} \sim \begin{cases} \mathrm{N}(0, \sigma^2) & \text{with probability } 1 - \alpha, \\ \mathrm{N}(0, k^2\sigma^2) & \text{with probability } \alpha, \end{cases} \tag{6.1.1}$$

for $i = 1, \ldots, n$, and $k > 1$. The mixture distribution (6.1.1) indicates that there exists a probability α of each observation being generated from an alternative distribution with the higher variance $k^2\sigma^2$. Verdinelli & Wasserman (1991) later on introduce a set of dummy variables $\boldsymbol{\delta} = (\delta^{(1)}, \ldots, \delta^{(n)})^{\mathrm{T}}$ in this model so that

$$Y^{(i)} \sim \begin{cases} \mathrm{N}((\boldsymbol{x}^{(i)})^{\mathrm{T}}\boldsymbol{\beta},\, k^2\sigma^2), & \text{if} \quad \delta^{(i)} = 1, \\ \mathrm{N}((\boldsymbol{x}^{(i)})^{\mathrm{T}}\boldsymbol{\beta},\, \sigma^2), & \text{if} \quad \delta^{(i)} = 0, \end{cases}$$

where $\boldsymbol{x}^{(i)}$ is the $p \times 1$ vector of the i^{th} row of X. Therefore, given $\boldsymbol{\delta}$, $\boldsymbol{Y} \sim \mathrm{N}_n(X\boldsymbol{\beta},\, \sigma^2 V)$, where V is a diagonal $n \times n$ matrix with $V_{ii} = (1 + \delta^{(i)}(k^2 - 1))$. Correspondingly, the likelihood of the data $\boldsymbol{Y}$ given the parameters $\boldsymbol{\beta}$, σ^2, α and $\boldsymbol{\delta}$ is given by

$$P(\boldsymbol{y}|\boldsymbol{\beta}, \sigma^2, \alpha) = \frac{1}{\sigma^n |V|^{1/2}} \exp\left\{ -\frac{1}{2\sigma^2} (\boldsymbol{y} - X\boldsymbol{\beta})^{\mathrm{T}} V^{-1} (\boldsymbol{y} - X\boldsymbol{\beta}) \right\}.$$

We shall now describe the priors for the parameters. We assume that the value of k is fixed and that α has a Beta (γ_1, γ_2) distribution with density

$$P_\alpha(\alpha) = \frac{\Gamma(\gamma_1 + \gamma_2)}{\Gamma(\gamma_1)\,\Gamma(\gamma_2)} \alpha^{\gamma_1 - 1} (1 - \alpha)^{\gamma_2 - 1}.$$

The expectation of this distribution is given by $\mathbb{E}(\alpha) = \gamma_1/(\gamma_1 + \gamma_2)$, which we shall denote by α_0. For $i = 1, \ldots, n$, the $\delta^{(i)}$ are independent with

$$\delta^{(i)} = \begin{cases} 1 & \text{with probability } \alpha, \\ 0 & \text{with probability } 1 - \alpha. \end{cases}$$

Therefore the probability of $\boldsymbol{\delta}$ given α is given by

$$P_{\delta|\alpha}(\boldsymbol{\delta}|\alpha) = \alpha^{\sum \delta_i} (1 - \alpha)^{n - \sum \delta_i}.$$

We assume the standard choice of improper prior distribution for β and σ^2 $P_{\beta,\sigma^2}(\beta, \sigma^2) = \sigma^{-2}$. Then the joint prior distribution for the parameters $\boldsymbol{\beta}$, σ^2, α and $\boldsymbol{\delta}$ is the product of $P_\alpha(\alpha)$, $P_{\delta|\alpha}(\boldsymbol{\delta}|\alpha)$ and $P_{\beta,\sigma^2}(\beta, \sigma^2)$.

The posterior distribution of the parameters $\boldsymbol{\beta}$, σ^2, α and $\boldsymbol{\delta}$ given the data $\boldsymbol{Y}$ is the product of the likelihood of the data and the joint prior distribution for all the parameters, suitably standardized. We shall refer to the standardized version of this posterior distribution as $\pi(\boldsymbol{\beta}, \sigma^2, \alpha, \boldsymbol{\delta})$, where

$$\begin{aligned} \pi(\boldsymbol{\beta}, \sigma^2, \alpha, \boldsymbol{\delta}) \;&=\; c\, \alpha^{\gamma_1 + \sum \delta^{(i)} - 1} (1 - \alpha)^{\gamma_2 + n - \sum \delta^{(i)} - 1} \frac{1}{\sigma^{2(\frac{n}{2} + 1)} |V|^{1/2}} \cdot \\ &\qquad \exp\left\{ -\frac{1}{2\sigma^2} (\boldsymbol{y} - X\boldsymbol{\beta})^{\mathrm{T}} V^{-1} (\boldsymbol{y} - X\boldsymbol{\beta}) \right\} \end{aligned} \tag{6.1.2}$$

for the appropriate value of c to make this a probability distribution. We shall make inferences by sampling from this posterior distribution. We can do this by using the Gibbs sampler (Geman & Geman (1984)).

Using equation (6.1.2) we can find the conditional distribution of each of the parameters $\boldsymbol{\beta}$, σ^2, α, and $\boldsymbol{\delta}$ given the others. Note that if $Z \sim \text{Gamma}\,(a, b)$, the density of its distribution is

$$\frac{b^a}{\Gamma(a)} z^{a-1} e^{-bz}$$

and its expectation is $a/(a+b)$. Then for $\hat{\boldsymbol{\beta}} = (X^{\mathrm{T}}V^{-1}X)^{-1}X^{\mathrm{T}}V^{-1}\boldsymbol{y}$, $\phi = 1 - 1/k^2$, and $r = \sum_i \delta^{(i)}$, the conditional distribution for each parameter is given as

$$\boldsymbol{\beta}|\sigma^2, \alpha, \boldsymbol{\delta} \sim \mathrm{N}_p(\hat{\boldsymbol{\beta}},\, \sigma^2(X^{\mathrm{T}}V^{-1}X)^{-1}), \tag{6.1.3}$$

$$\sigma^{-2}|\boldsymbol{\beta}, \alpha, \boldsymbol{\delta} \sim \text{Gamma}\,(\tfrac{n}{2},\, \tfrac{1}{2}(\boldsymbol{y} - X\boldsymbol{\beta})^{\mathrm{T}}V^{-1}(\boldsymbol{y} - X\boldsymbol{\beta})), \tag{6.1.4}$$

$$\alpha|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta} \sim \text{Beta}\,(\gamma_1 + r,\, \gamma_2 + n - r), \tag{6.1.5}$$

and the $\delta^{(i)}$ are conditionally independent given $\boldsymbol{\beta}$, σ^2 and α with

$$\delta^{(i)}|\boldsymbol{\beta}, \sigma^2, \alpha \sim \text{Bernoulli}\left(\left[1 + \tfrac{1-\alpha}{\alpha}\, k\, \exp\left\{-\tfrac{\phi(y^{(i)} - (x^{(i)})^{\mathrm{T}}\beta)^2}{2\sigma^2}\right\}\right]^{-1}\right),$$
$$i = 1, \ldots, n. \tag{6.1.6}$$

A single cycle of the Gibbs sampler, i.e., one step of the Markov chain, consists of updating the parameters $(\boldsymbol{\beta}, \sigma^2, \alpha, \boldsymbol{\delta})$ one at a time using their conditional distributions (6.1.3) to (6.1.6). The probability of observation i being an outlier in this model can be estimated using the proportion of iterations that $\delta^{(i)} = 1$.

6.2 Example 5: Star cluster CYG OB1

We look at a real data set: the star cluster CYG OB1. Two variables are observed in $n = 47$ stars in the direction of Cygnus. The independent variable (X) is the logarithm of the effective temperature at the surface of the stars and the dependent variable (Y) is the logarithm of the light intensity. The values are provided by Rousseeuw & Leroy (1987), and are given in Appendix B. A scatter plot of the data is shown in Figure 6-1. From this plot we could see there are four observations that look to be outliers. "Masking" could be an issue here, as these four observations might not appear to be outliers when considered individually, but may do so when all four observations are considered together as possible outliers. We shall investigate application of the normal scale contamination model to this data set.

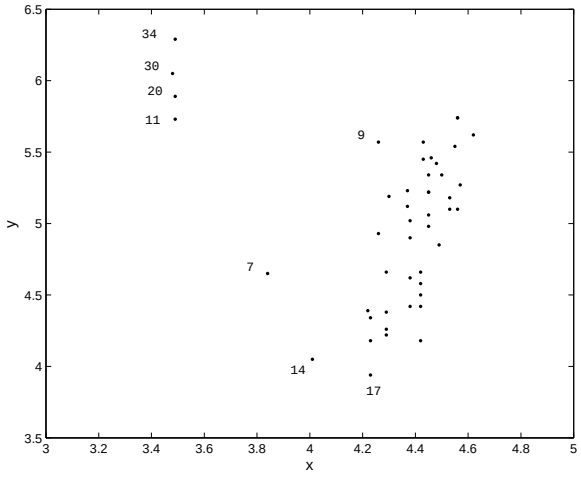


Figure 6-1: A scatter plot of the star cluster CYG OB1 data set.

For the priors, we choose $\gamma_1 = 5$ and $\gamma_2 = 42$ so that $\alpha_0 \approx 0.1$. Here we followed Justel & Peña (2001) by setting $\gamma_1+\gamma_2 = n$, which implies $\mathbb{E}(\alpha|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta}) = \alpha_0/2+r/(2n)$ since $\alpha|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta} \sim \text{Beta}\,(\gamma_1 + r,\, \gamma_2 + n - r)$. We also choose $k = 5$. Therefore the task for an MCMC sampler in this example is to sample from the distribution $\pi(\boldsymbol{\beta}, \sigma^2, \alpha, \boldsymbol{\delta})$ as given by equation (6.1.2) with these values of γ_1, γ_2 and k.

6.3 Sampling the target distribution of Example 5 using the Gibbs sampler

In order to sample from the target distribution $\pi(\boldsymbol{\beta}, \sigma^2, \alpha, \boldsymbol{\delta})$ defined by equation (6.1.2), we ran the Gibbs sampler for 100,000 iterations. We estimate the posterior probability of each observation being an outlier by taking the sample mean of $\delta^{(i)}$, i.e., the proportion of iterations that $\delta^{(i)} = 1$. The estimated values of this posterior probability for each observation $i = 1, \ldots, 47$ are shown in Figure 6-2. From this figure we can see that none of the observations stood out as being an outlier. This result is surprising since from the scatter plot of the data shown in Figure 6-1 we could see there are four observations that look to be substantial outliers.

This data set was discussed by Justel & Peña (1996), where they also showed that the Gibbs sampler did not identify any outliers in this data set after 10,000 iterations. Justel & Peña (1996) point out that this is because the posterior distribution arising from the normal scale contamination model can be multi-modal, and the Gibbs sampler can take a long time to move between different modes, especially in data sets with masking problems.

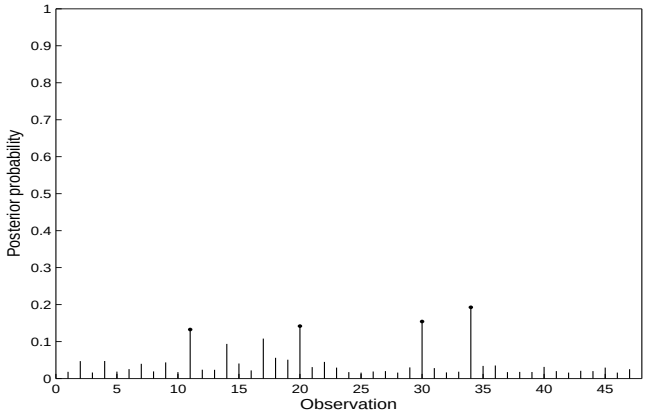


Figure 6-2: The posterior probability of each observation in the star cluster CYG OB1 being an outlier, obtained using the Gibbs sampler.

Justel & Peña (1996) believe that "when the set of data has many outliers that mask each other, Gibbs sampling will fail and posterior distributions are poorly estimated". Corresponding to this belief, Justel & Peña (1996) and Justel & Peña (2001) have suggested methods to overcome the problem that the Gibbs sampler faces in detecting masking outliers. Justel & Peña (1996) suggest a method for finding any subsets of masked outliers using repeated runs of the Gibbs sampler. They observe that if the Gibbs sampler is repeatedly run starting in a randomly chosen state where most of the observations are labelled as non-outliers and only a few labelled as outliers, then it will converge randomly to one of the modes. Justel & Peña (2001) later on propose a solution based on a careful initialization of the Gibbs sampler with a good candidate set of outliers. This set of outliers is identified through some initial parallel runs of the Gibbs sampler.

However, although the methods proposed by Justel & Peña (1996) and Justel & Peña (2001) may find the important modes through careful initialization, their Gibbs sampler does not move well between modes. Therefore, if the posterior distribution that arises from the model is actually multi-modal, they do not correctly assess the probability of subsets of the modes, corresponding to different sets of observations being outliers in the model. We shall take a different approach, applying our mode jumping approach to sample effectively from a set of modes in the posterior distribution.

6.4 Applying the mode jumping approach to the outlier regression model

6.4.1 General methodology

We shall apply our mode jumping approach to the outlier regression model, using the MJM method with implicit modelling. Define the parameter vector to be $\boldsymbol{Z} = (\beta^{(0)}, \ldots, \beta^{(p-1)}, \sigma^2, \alpha, \delta^{(1)}, \ldots, \delta^{(n)})$, of length $p' = p+2+n$. Let $Z^{(i)}$ represent the i^{th} component of $\boldsymbol{Z}$ while $Z^{(-i)} = \{Z^{(j)} : j \neq i\}$. Correspondingly $\pi(\boldsymbol{z}) = \pi(\boldsymbol{\beta}, \sigma^2, \alpha, \boldsymbol{\delta})$. To apply the MJM method, we go through the three stages:

1. Initial exploration, where we try to find the locations of all the modes;
2. Clustering, where we cluster the mode locations and find a single mode location to represent each cluster;
3. Sampling, where we use the mode locations from the clustering stage to jump between the modes.

Initial exploration

To find all mode locations, we use simulated annealing to optimize $\pi(\boldsymbol{z})$, which is equivalent to running the Gibbs sampler with conditional distributions raised to the power $1/T(k)$. This is because simulating from a target distribution proportional to $\pi(x)^{1/T(k)}$ is equivalent to simulating from distributions proportional to $\{\pi_{Z^{(i)}|Z^{(-i)}}(z^{(i)}|z^{(-i)})\}^{1/T(k)}$, $i = 1, \ldots, p'$ instead, as shown in section 5.4.1. Therefore on the k^{th} cycle we simulate parameters from the distributions

$$\boldsymbol{\beta}_{T(k)}|\sigma^2, \alpha, \boldsymbol{\delta} \sim \mathrm{N}_p(\hat{\boldsymbol{\beta}},\, \sigma^2 T(k)(X^{\mathrm{T}}V^{-1}X)^{-1}),$$

$$\sigma^{-2}_{T(k)}|\boldsymbol{\beta}, \alpha, \boldsymbol{\delta} \sim \text{Gamma}\,(\tfrac{\frac{n}{2}+1}{T(k)} - 1,\, \tfrac{1}{2\,T(k)}(\boldsymbol{y} - X\boldsymbol{\beta})^{\mathrm{T}}V^{-1}(\boldsymbol{y} - X\boldsymbol{\beta})),$$

$$\alpha_{T(k)}|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta} \sim \text{Beta}\,(\tfrac{\gamma_1+r-1}{T(k)} + 1,\, \tfrac{\gamma_2+n-r-1}{T(k)} + 1),$$

$$\delta^{(i)}_{T(k)}|\boldsymbol{\beta}, \sigma^2, \alpha \sim \text{Bernoulli}\left(\left[1 + \left(\tfrac{1-\alpha}{\alpha}\, k\, \exp\left\{-\tfrac{\phi(y^{(i)} - (x^{(i)})^{\mathrm{T}}\beta)^2}{2\sigma^2}\right\}\right)^{\frac{1}{T(k)}}\right]^{-1}\right), \quad i = 1, \ldots, n.$$

After running the Gibbs sampler with simulated annealing to a conclusion, we apply a "hill-climbing" process to find the point with maximum $\pi(z)$ in this vicinity. In this hill-climbing process, we maximize $\pi(z)$ for one variable with the other variables fixed, i.e., we maximize the conditional distributions $\pi_{Z^{(i)}|Z^{(-i)}}(z^{(i)}|z^{(-i)})$, $i = 1, \ldots, p'$. For $\delta^{(i)}$, $i = 1, \ldots, n$, we update its value to 1 if the conditional probability of $\delta^{(i)} = 1$ is greater than 0.5. For $\boldsymbol{\beta}$, σ^2 and α, we update their values to the modal value of each conditional distribution, for example we change $\boldsymbol{\beta}$ to $\hat{\boldsymbol{\beta}}$, which is the mode (and mean)

of its conditional normal distribution given $\boldsymbol{\delta}$ (which appears in V). We repeat the process of updating the values of $\boldsymbol{\delta}$, $\boldsymbol{\beta}$, σ^2 and α until the values converge. Using a new set of starting values each time, we repeat the procedure of simulated annealing and hill-climbing l times to obtain l mode locations, which we denote by $\boldsymbol{Z}_j$, $j = 1, \ldots, l$.

Clustering

We define the "distance" between two mode locations $\boldsymbol{Z}_i$ and $\boldsymbol{Z}_j$ to be their scaled Euclidean distance as given by equation (4.1.1) in section 4.1.1, using the scaling values as defined in the same section. Using this definition of distance, we cluster the l possible mode locations that we found in the initial exploration stage using the hierarchical clustering algorithm as described in section 4.1.1. The clustering algorithm then produces a number, m, of distinct mode locations.

Sampling

We define the "nearest mode" function, $h(\boldsymbol{z}) \in \{1, \ldots, m'\}$, so that it will choose the nearest mode j which has the minimum scaled Euclidean distance between $\boldsymbol{Z}$ and η_j, again using the scaling values as defined in section 4.1.1, but only using the m mode locations found from the clustering stage. The sampling stage then proceeds as described in section 3.3.4, where the mode jumping step follows the MJM method with implicit modelling, therefore using a cycle of Gibbs sampling as the final step in generating a proposal. Note that here we shall use $\boldsymbol{z}_q$ to denote the proposal state instead of $\boldsymbol{y}$; we define the probability of moving from a mode location $\boldsymbol{\eta}_j$ to a proposal state $\boldsymbol{z}_q$ via one cycle of the Gibbs sampler to be

$$q_j(\boldsymbol{z}_q) \;=\; \prod_{i=1}^{p'} \pi_{Z^{(i)}|Z^{(-i)}}(z_q^{(i)} | z_q^{(1)}, \ldots, z_q^{(i-1)}, \eta_j^{(i+1)}, \ldots, \eta_j^{(p')}).$$

6.4.2 Application to Example 5

We apply our MJM method to sample from the target distribution $\pi(\boldsymbol{\beta}, \sigma^2, \alpha, \boldsymbol{\delta})$ defined by equation (6.1.2), and compare the performance of this method with that of the Gibbs sampler seen in section 6.3.

Implementation

1. Initial exploration

   We carried out a run of length 200 using the Gibbs sampler with simulated annealing, using the logarithmic temperature function, $T(k) = 1/\ln(1+k)$, and applied the hill-climbing process to the end values. Here the hill-climbing process took only one iteration to converge. We repeated the procedure of simulated annealing and hill-climbing 100 times. In each run we used the starting values $\boldsymbol{\beta}$

Mode location	Values			
	$\boldsymbol{\beta}$	σ^2	α	Observations i where $\delta^{(i)} = 1$
$\boldsymbol{\eta}_1$	(6.793, –0.413)	0.293	0.043	–
$\boldsymbol{\eta}_2$	(–2.325, 1.654)	0.162	0.087	11, 20, 30, 34

Table 6.1: Values of the mode locations $\boldsymbol{\eta}_i$, $i = 1, 2$, for the star cluster CYG OB1 data.

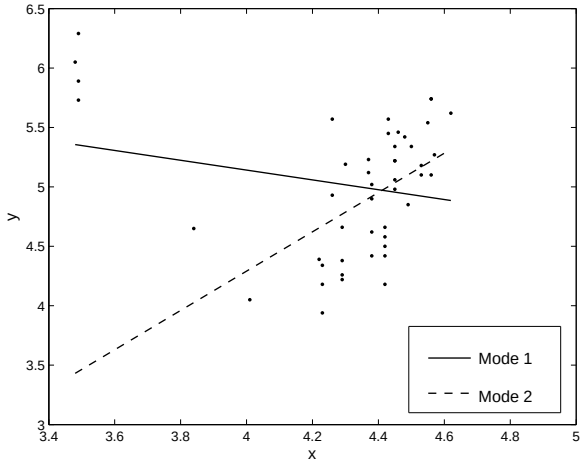


Figure 6-3: A scatter plot of the star cluster CYG OB1 data set with the fitted regression lines for mode locations $\boldsymbol{\eta}_1$ and $\boldsymbol{\eta}_2$.

$= (0,0)$, $\sigma^2 = 1$ and simulated α from the prior distribution Beta $(\gamma_1 = 5, \gamma_2 = 42)$. For $\boldsymbol{\delta}$, in 50 runs we used the starting values $\boldsymbol{\delta}_s$ where $\sum_i (1 - \delta_s^{(i)}) = 3$ and the three values of i where $\delta_s^{(i)} = 0$ are uniformly chosen. Here we followed the starting values for $\boldsymbol{\delta}$ as suggested by Sharp (2003, chapter 5), who argues that a particular mode can be found by starting the Gibbs sampler in a state where $\boldsymbol{\delta}$ consists of mostly ones (outliers) except for a few zeroes (non-outliers), where the non-outliers agree with the mode. In the other 50 runs we used the starting values $\boldsymbol{\delta}_s$ where $\sum_i (1 - \delta_s^{(i)}) = 44$ and the three values of i where $\delta_s^{(i)} = 1$ are uniformly chosen. Since the parameters are updated in the order of $\boldsymbol{\beta}$, σ^2, α and lastly $\boldsymbol{\delta}$, therefore our choice of starting values should allow separate runs of the initial exploration to reach different modes. These runs then produced 100 mode locations, only 2 of which were distinct.

2. Clustering

   For the clustering algorithm, we chose $\xi = 3.0$ and obtained $m = 2$. The values of the corresponding mode locations $\boldsymbol{\eta}_i$, $i = 1, 2$, are given in Table 6.1. The fitted regression lines using the values of these mode locations are shown in Figure 6-3.

3. Sampling

In the sampling stage, we combine local movements using the Gibbs sampler and mode jumping steps: at each iteration we choose to do Gibbs sampling with probability θ and to do a mode jumping step with probability $1-\theta$. The mode jumping step proceeds as given in section 6.4.1, where we choose to jump to a different mode with equal probability, i.e., $p_{ij} = 1$ for $j \neq i$.

Results for Example 5

We want to compare sampling from the target distribution $\pi(\boldsymbol{z})$ using the MJM method with sampling using the Gibbs sampler. When comparing any sampling methods we continue to use the comparison criteria described in section 4.2.2. In doing this, we define the nearest mode function $m(\boldsymbol{z})$ so that $m(\boldsymbol{z}) = 1$ if the sum of the $\delta^{(i)}$ for $i = 11, 20, 30, 34$ is 2 or less and $m(\boldsymbol{z}) = 2$ if otherwise. This is because modes are defined primarily by $\boldsymbol{\delta}$, as for a given value of $\boldsymbol{\delta}$ the function π is unimodal in $\boldsymbol{\beta}$, σ^2 and α. Therefore, for simplicity we define the "nearest mode" using only $\boldsymbol{\delta}$. Also, we choose to classify cases with two of the four $\delta^{(i)}$ equal to 1 as mode 1 so that mode 2 corresponds to all or nearly all of the 4 observations in question labelled as outliers.

We simulated a chain of length 100,000 using the MJM method and the Gibbs sampler, using the same starting point for both methods. For our mode jumping method, we choose $\theta = 0.0$, 0.25, 0.5, and 0.75. The values of $m(\boldsymbol{z})$ for the first 1,000 iterations of the various simulated chains are shown in Figure 6-4. From this figure, we can see that the chain simulated using the Gibbs sampler is doing the worst, as it is not mixing well. The estimates from these 100,000 iterations of the transition matrices for the movement of the chain between the two modes for these two methods are given by

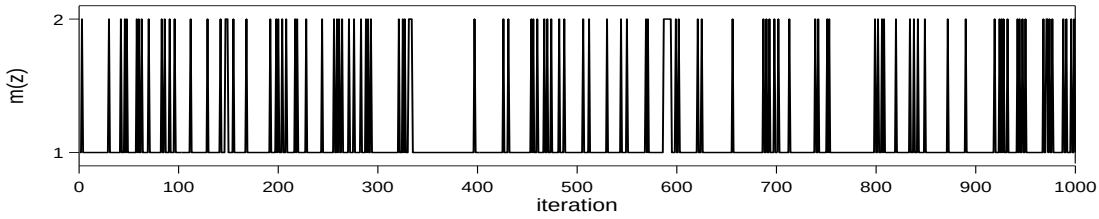
$$\widehat{P}_{gs} = \begin{pmatrix} 0.997 & 0.003 \\ 0.021 & 0.979 \end{pmatrix}$$

for the Gibbs sampler, and

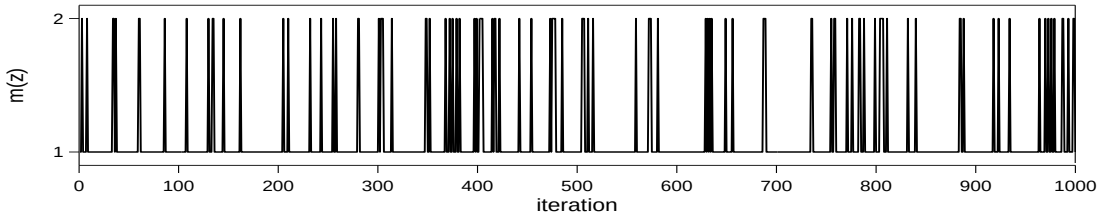
$$\widehat{P}_{m} = \begin{pmatrix} 0.907 & 0.093 \\ 0.684 & 0.316 \end{pmatrix}$$

for the MJM method with $\theta = 0.25$. We can see that there are much larger values off the diagonal in $\widehat{P}_m$.

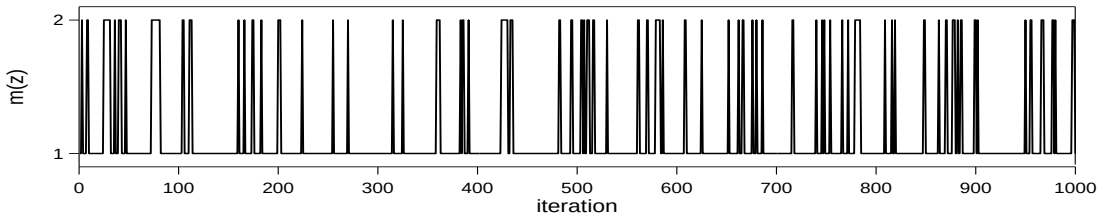
The overall results of the MJM method and the Gibbs sampler are summarized in Table 6.2. From Table 6.2, we can see that the MJM method yields much higher mode jumping rates than the Gibbs sampler. This can also be observed from Figure 6-4. In fact, for the case where $\theta = 0.0$ our mode jumping proposals will successfully jump between modes around 44 times as frequently as the proposals in the Gibbs sampler.



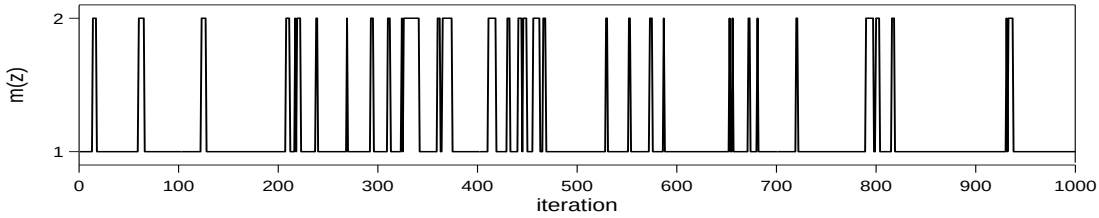
(a)(i) the MJM method with $\theta = 0.0$



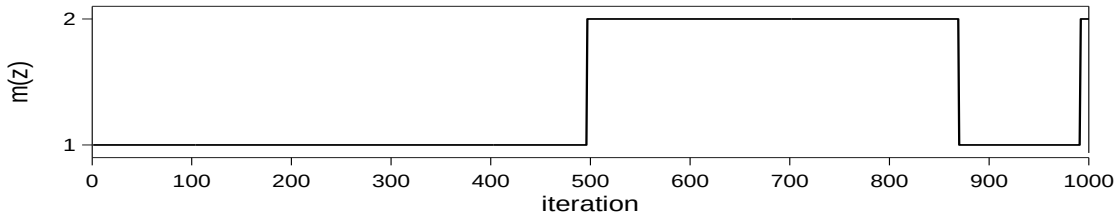
(a)(ii) the MJM method with $\theta = 0.25$



(a)(iii) the MJM method with $\theta = 0.5$



(a)(iv) the MJM method with $\theta = 0.75$



(b) the Gibbs sampler

Figure 6-4: Plots of the values of $m(\boldsymbol{z})$ for the first 1,000 iterations, where the chains are simulated using (a) the MJM method with various values of θ; and (b) the Gibbs sampler.

	Method				
	MJM				Gibbs
	$\theta = 0.0$	$\theta = 0.25$	$\theta = 0.5$	$\theta = 0.75$	sampler
Mode jumping rate (%)	21.99	16.43	11.23	5.80	0.49
Total computation ($\times 10^5$)					
a) initial exploration					
i) Gibbs cycles	0.2	0.2	0.2	0.2	0
b) sampling					
i) Gibbs cycles	2	1.75	1.5	1.25	1
ii) evaluations of $\pi(\boldsymbol{z}_q)/\pi(\boldsymbol{z})$	1	0.75	0.5	0.25	0
iii) total	3	2.5	2	1.5	1
Eigenvalue λ_1 from $\widehat{P}$	-0.028	0.223	0.466	0.713	0.976
Estimated IAC for $f^{(1)} = I\{m(\boldsymbol{z}) = 1\}$ calculated using $\widehat{P}$					
$\hat{\tau}(f^{(1)})$	0.95	1.57	2.75	6.13	81.60

Table 6.2: Overall results for a chain of run length 100,000, simulated using the MJM method and the Gibbs sampler.

We also have to compare the computation used by each method. In this case, there are two types of computation used: 1) the computation used in one cycle of the Gibbs sampler update; and 2) the computation used to evaluate $\pi(\boldsymbol{z}_q)/\pi(\boldsymbol{z})$ in the acceptance probability $\alpha(\boldsymbol{z}, \boldsymbol{z}_q)$. The Gibbs sampler uses one cycle of the Gibbs sampler update in each iteration. In the case of the MJM method, in each iteration we choose to do Gibbs sampling with probability θ and to do a mode jumping step with probability $1 - \theta$, where for each mode jumping step we are using two cycles of the Gibbs sampler update and one evaluation of $\pi(\boldsymbol{z}_q)/\pi(\boldsymbol{z})$. We note that one evaluation of $\pi(\boldsymbol{z}_q)/\pi(\boldsymbol{z})$ is roughly equal to one cycle of the Gibbs sampler update, as discussed in section 5.4.2. This means that one mode jumping step costs 3 times as much as one cycle of the Gibbs sampler update. In fact, the MJM method with $\theta = 0.0$ costs 3 times as much as the Gibbs sampler.

From Table 6.2, we can see that values of λ^*, where $\lambda^* = |\lambda_1|$, for the MJM method are lower than for the Gibbs sampler, therefore convergence to the correct distribution over the two modes is relatively faster for the MJM method. Note that for the MJM method with $\theta = 0$, the value of λ_1 is negative. According to Green & Han (1992), rapid weak convergence to equilibrium is obtained by having eigenvalues, except for $\lambda_0 = 1$, which are small in absolute value, whilst good asymptotic mean squared error of estimation is helped by having negative eigenvalues.

The values of the IAC, $\tau(f^{(1)})$, for $f^{(1)}(\boldsymbol{z}) = I\{m(\boldsymbol{z}) = 1\}$, which are estimated

using the corresponding $\widehat{P}$ of each method, are also given in Table 6.2. We do not report $\tau(f^{(2)})$ since $f^{(2)}(\boldsymbol{z}) = 1 - f^{(1)}(\boldsymbol{z})$ and, therefore, $\tau(f^{(2)}) = \tau(f^{(1)})$. We can see that the values of $\hat{\tau}(f^{(1)})$ for the MJM method are lower than that for the Gibbs sampler. In particular, the value of $\hat{\tau}(f^{(1)})$ for the MJM method using $\theta = 0.0$ is less than one eighty-fifth of the value for the Gibbs sampler, i.e., our method obtains the same estimation accuracy as the Gibbs sampler using relatively just eighty-fifth of the run length. This means that the MJM method has better estimation performance than the Gibbs sampler. However, we have to take into account the cost of each method. If we combine this with the fact that the MJM method with $\theta = 0.0$ costs 3 times as much as the Gibbs sampler, then overall this method is still 28 times more efficient than the Gibbs sampler.

We can estimate the posterior probability that $\delta^{(i)} = 1$ for $i = 11, 20, 30, 34$ by estimating the weight associated with mode 2. From the 100,000 iterations of the MJM chain with any value of θ, we found the proportion of time the chain spent in mode 2 is around 0.12. Therefore the posterior probabilities given in Figure 6-2 are actually quite reasonable. This means that even though the four observations look to be "strange", the model does not classify them as clear outliers.

In general, the MJM method performs more efficiently than the Gibbs sampler when applied to this example. Our mode jumping approach does have extra set up costs, which need to be considered in efficiency comparisons. In the initial exploration stage, our method uses 1 Gibbs sampler update for each iteration in the simulated annealing. However, with the extent of initial exploration used in this example, the total amount of computation used in the initial exploration stage is just a small fraction of the amount used in the sampling stage; for example, for the case where $\theta = 0$ the computation used in the initial exploration stage is only 6.7% of the amount used in the sampling stage. This means that the initial exploration is relatively cheap and accounting for it makes little difference to the efficiency comparisons noted above. In our initial exploration, we do multiple short runs, using specific starting values that should allow separate runs of the initial exploration to reach different modes. This would ensure that the probability of missing a mode is low. We could also increase the number of multiple short runs in the initial exploration stage substantially to reduce the probability of missing a mode, with little effect on the total computation.

We have also simulated from the posterior distribution using other values of γ_1, γ_2 and k in the Bayesian model. Comparisons of the MJM method versus the Gibbs sampler show similar features for these other cases. For example, when using $\alpha_0 = 0.2$ and $k = 5$ the MJM method still performs better than the Gibbs sampler as its mode jumping rate is nearly 39 times higher and it is 23 times more efficient. If we use higher

values of k, for example $k = 8$, the MJM method's mode jumping rate is around 140 times higher than that for the Gibbs sampler and it is around 60 times more efficient.

6.5 Sampling from the marginal posterior distribution of $\boldsymbol{\delta}$

An alternative way to identify outliers in the Bayesian linear regression using the normal scale contamination model is to sample from the *marginal* posterior distribution of $\boldsymbol{\delta}$ given $\boldsymbol{Y}$ by integrating over $\boldsymbol{\beta}$, σ^2 and α in the posterior distribution of $(\boldsymbol{\beta}, \sigma^2, \alpha, \boldsymbol{\delta})$ given $\boldsymbol{Y}$. We show in Appendix C that the marginal posterior distribution of $\boldsymbol{\delta}$ given $\boldsymbol{Y}$ is given by

$$P(\boldsymbol{\delta}|\boldsymbol{y}) \propto \left(\frac{|(X^{\mathrm{T}}V(\delta)^{-1}X)^{-1}|}{|V(\delta)|}\right)^{1/2} \frac{\Gamma(\gamma_1 + \sum \delta^{(i)})\,\Gamma(\gamma_2 + n - \sum \delta^{(i)})}{\left[(\boldsymbol{y} - X\hat{\boldsymbol{\beta}}(\delta))^{\mathrm{T}}V(\delta)^{-1}(\boldsymbol{y} - X\hat{\boldsymbol{\beta}}(\delta))\right]^{\frac{(n-p)}{2}}}$$

where $V(\delta) = \mathrm{diag}(1 + \delta^{(i)}(k^2 - 1))$ and $\hat{\boldsymbol{\beta}}(\delta) = (X^{\mathrm{T}}V(\delta)^{-1}X)^{-1}X^{\mathrm{T}}V(\delta)^{-1}\boldsymbol{y}$. In this case, we wish to sample from $\pi(\boldsymbol{\delta})$, the standardized version of the posterior distribution, where

$$\pi(\boldsymbol{\delta}) = c\left(\frac{|(X^{\mathrm{T}}V(\delta)^{-1}X)^{-1}|}{|V(\delta)|}\right)^{1/2} \frac{\Gamma(\gamma_1 + \sum \delta^{(i)})\,\Gamma(\gamma_2 + n - \sum \delta^{(i)})}{\left[(\boldsymbol{y} - X\hat{\boldsymbol{\beta}}(\delta))^{\mathrm{T}}V(\delta)^{-1}(\boldsymbol{y} - X\hat{\boldsymbol{\beta}}(\delta))\right]^{\frac{(n-p)}{2}}}, \quad (6.5.1)$$

for the appropriate value of c to make this a probability distribution. We can sample from $\pi(\boldsymbol{\delta})$ using the Gibbs sampler, where we sample each $\delta^{(i)}$ from its conditional distribution $\pi_{\delta^{(i)}|\delta^{(-i)}}(\delta^{(i)}|\delta^{(-i)})$. When each $\delta^{(i)}$ has been updated in turn, for $i = 1, \ldots, n$, then a single cycle of the Gibbs sampler is completed. However, we expect our mode jumping methods will provide a more efficient sampling mechanism.

6.6 Applying the MJM method to sample the marginal posterior distribution of $\boldsymbol{\delta}$

6.6.1 General methodology

We shall apply the MJM method to sample from the target distribution $\pi(\boldsymbol{\delta})$ defined by equation (6.5.1). In this case we define the parameter vector to be $\boldsymbol{Z} = (\delta^{(1)}, \ldots, \delta^{(n)})$, a vector of length n, and correspondingly our target distribution is $\pi(\boldsymbol{z}) = \pi(\boldsymbol{\delta})$. To apply the MJM method we shall go the three stages: 1) initial exploration; 2) clustering; and 3) sampling. While the clustering and sampling stages remain the same as described in section 6.4.1, the initial exploration stage needs to be defined differently.

Mode location	Observation i where $\eta^{(i)} = \delta^{(i)} = 1$
$\boldsymbol{\eta}_1$	–
$\boldsymbol{\eta}_2$	11, 20, 30, 34

Table 6.3: Values of the mode locations $\boldsymbol{\eta}_i$, $i = 1, 2$ for the star cluster CYG OB1 data.

Initial exploration

To find all mode locations, we use the Gibbs sampler with simulated annealing. We simulate, on the k^{th} cycle/iteration and at the i^{th} component, from the distribution proportional to

$$\left\{\pi_{Z^{(i)}|Z^{(-i)}}(z^{(i)}|z^{(-i)})\right\}^{1/T(k)}.$$

After running the Gibbs sampler with simulated annealing to a conclusion, we apply the hill-climbing process to the end values where we update $Z^{(i)}$, $i = 1, \ldots, n$, by changing the value to 1 if the conditional probability of $Z^{(i)} = 1$ is greater than 0.5. We repeat the process of updating the values of $Z^{(i)}$ until the values converge. Using a different set of starting values each time, we repeat the procedure of simulated annealing and hill-climbing l times to obtain l mode locations.

6.6.2 Application to Example 5

We apply the MJM method to Example 5 and compare the performance of our method with that of the Gibbs sampler.

Implementation

1. Initial exploration
   We carried out a run of length 200 using the Gibbs sampler with simulated annealing, using the logarithmic temperature function, $T(k) = 1/\ln(1+k)$, with hill-climbing on termination. We repeated this procedure 100 times. For 50 runs we used the starting values $\boldsymbol{z} = \boldsymbol{\delta}_s$ where $\sum_i (1 - \delta_s^{(i)}) = 3$ and the three values of i where $\boldsymbol{z}\delta_s^{(i)} = 0$ are uniformly chosen. For the other 50 we used the starting values $\boldsymbol{\delta}_s$ where $\sum_i (1 - \delta_s^{(i)}) = 44$ and the three values of i where $\delta_s^{(i)} = 1$ are uniformly chosen. These runs then produced 100 mode locations, only two of which were distinct.

2. Clustering
   For the clustering algorithm, we chose $\xi = 1.0$ and obtained $m = 2$. The values of the corresponding mode locations $\boldsymbol{\eta}_i$, $i = 1, 2$, are given in Table 6.3.

3. Sampling
   We combine local movements using the Gibbs sampler and mode jumping steps: at each iteration we choose to do Gibbs sampling with probability θ and to do a

mode jumping step with probability $1-\theta$. The mode jumping step proceeds as given in section 6.6.1, where we choose to jump to a different mode with equal probability, i.e., $p_{ij} = 1$ for $j \neq i$.

Results for Example 5

We simulated a chain of length 100,000 using the MJM method and the Gibbs sampler, using the same starting point for both methods. For our mode jumping method, we choose $\theta = 0.0$, 0.25, 0.5, and 0.75. The values of $m(\boldsymbol{z})$ for the first 1,000 iterations of the various simulated chains are shown in Figure 6-5. From this figure, we can see that mixing is better for the lower values of θ and the chain simulated using the Gibbs sampler is not mixing at all well. The estimate from these 100,000 iterations of the transition matrices for the movement of the chain between the two modes for these two methods are given by

$$\widehat{P}'_{gs} = \begin{pmatrix} 0.9942 & 0.0058 \\ 0.0481 & 0.9519 \end{pmatrix}$$

for the Gibbs sampler, and

$$\widehat{P}'_{m} = \begin{pmatrix} 0.8967 & 0.1033 \\ 0.7459 & 0.2541 \end{pmatrix}$$

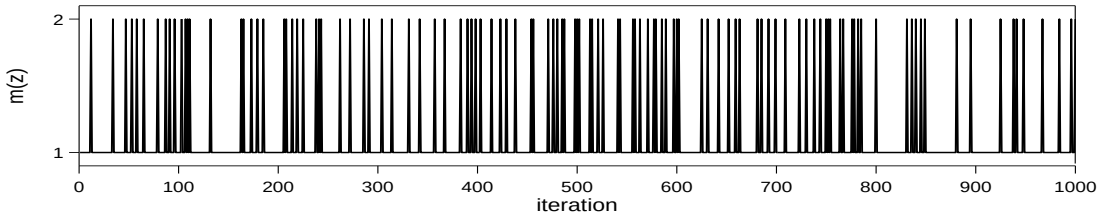
for the MJM method with $\theta = 0.25$. We can see that there are larger values off the diagonal in $\widehat{P}'_m$.

The overall results of both methods are summarized in Table 6.4. We can see that our MJM method yields much higher mode jumping rates than the Gibbs sampler. This can also be observed from Figure 6-5. In fact, for the case where $\theta = 0$ our mode jumping proposals will successfully jump between modes more than 22 times as frequently as the proposals in the Gibbs sampler.

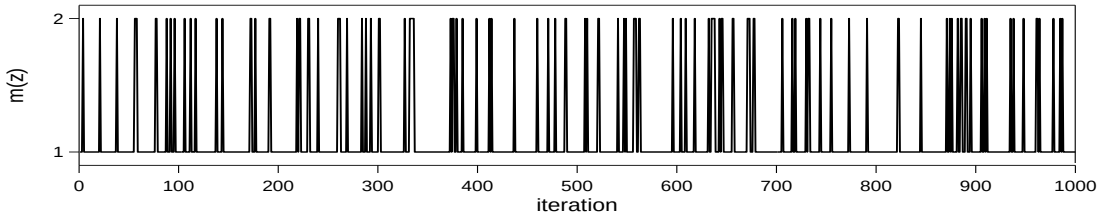
We also have to compare the computation used by each method. As in section 6.4.2, in each iteration the Gibbs sampler uses one cycle of the Gibbs sampler update while the MJM method uses $(2-\theta)$ cycles of the Gibbs sampler update and $(1-\theta)$ evaluations of $\pi(\boldsymbol{z}_q)/\pi(\boldsymbol{z})$. We note that one evaluation of $\pi(\boldsymbol{z}_q)/\pi(\boldsymbol{z})$ is equal to one Gibbs sampler update of a single element of $\boldsymbol{Z}$, as this involves evaluating the conditional probability of say, $Z^{(i)} = 1$, which is equal to

$$\frac{\pi(\boldsymbol{z}_1)}{\pi(\boldsymbol{z}_0) + \pi(\boldsymbol{z}_1)}$$

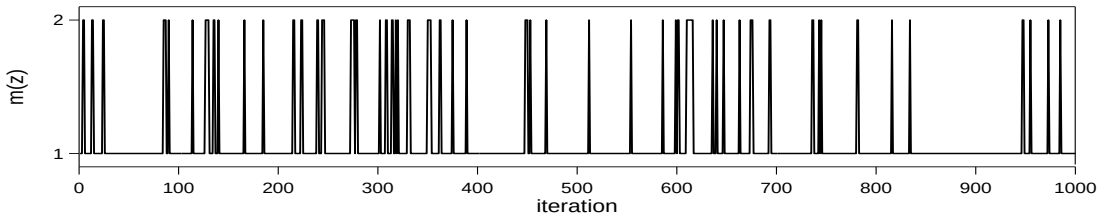
where $\boldsymbol{Z}_1$ is the vector $\boldsymbol{Z}$ with $Z^{(i)} = 1$ and $\boldsymbol{Z}_1$ the vector with $Z^{(i)} = 0$. This means that one mode jumping step costs equivalently $2 + 1/n = 2.021$ times as much



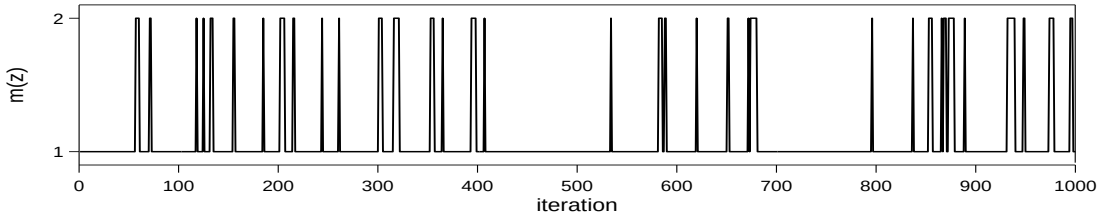
(a)(i) the MJM method with $\theta = 0.0$



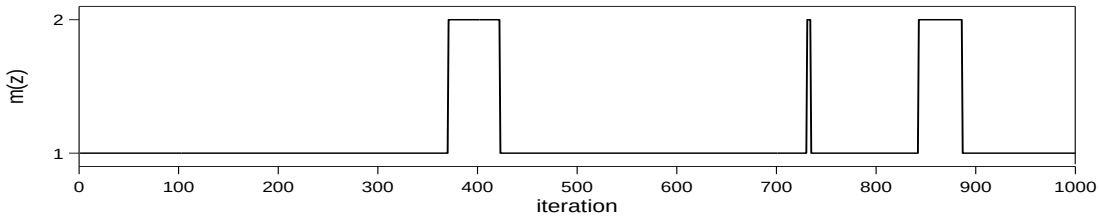
(a)(ii) the MJM method with $\theta = 0.25$



(a)(iii) the MJM method with $\theta = 0.5$



(a)(iv) the MJM method with $\theta = 0.75$



((b) the Gibbs sampler

Figure 6-5: Plots of the values of $m(\boldsymbol{z})$ for the first 1,000 iterations, where the chains are simulated using (a) the MJM method with various values of θ; and (b) the Gibbs sampler.

	Method				
	MJM				Gibbs
	$\theta = 0.0$	$\theta = 0.25$	$\theta = 0.5$	$\theta = 0.75$	sampler
Mode jumping rate (%)	23.56	18.14	12.47	6.82	1.03
Total computation ($\times 10^5$)					
a) initial exploration					
i) Gibbs cycles	0.2	0.2	0.2	0.2	0
b) sampling					
i) Gibbs cycles	2	1.75	1.5	1.25	1
ii) evaluations of $\pi(\boldsymbol{z}_q)/\pi(\boldsymbol{z})$	1	0.75	0.5	0.25	0
iii) total	2.02	1.77	1.51	1.26	1
Eigenvalue λ_1 of $\widehat{P}'$	-0.125	0.151	0.412	0.688	0.946
Estimated IAC for $f^{(1)} = I\{m(\boldsymbol{z}) = 1\}$ calculated using $\widehat{P}'$					
$\hat{\tau}(f^{(i)})$	0.78	1.36	2.40	5.41	35.98

Table 6.4: Overall results for chain of run length 100,000, simulated using the MJM method and the Gibbs sampler.

as one cycle of the Gibbs sampler update. Then the MJM method uses equivalently $(2.021 - 1.021\theta)$ Gibbs sampler updates. In fact, the MJM method with $\theta = 0.0$ costs 2.021 times as much as the Gibbs sampler.

From Table 6.4, we can see that values of λ^*, where $\lambda^* = |\lambda_1|$, for the MJM method are lower than for the Gibbs sampler, therefore convergence to the correct distribution over the two modes is faster for the MJM method. Since the value $\lambda^* = 0.946$ for the Gibbs sampler is close to one, overall the chain simulated using the Gibbs sampler is converging slowly. Note that for the MJM method with $\theta = 0$, the value of λ for which $|\lambda| = \lambda^*$ is negative.

The values of the IAC, $\tau(f)$, for $f^{(1)}(\boldsymbol{z}) = I\{m(\boldsymbol{z}) = 1\}$, which are estimated using the corresponding $\widehat{P}'$ of each method, are also given in Table 6.4. We can see that the values of $\hat{\tau}(f^{(1)})$ for the MJM method are lower than that for the Gibbs sampler. In particular, the value of $\hat{\tau}(f^{(1)})$ for the MJM method with $\theta = 0$ is less than one forty-sixth of the value for the Gibbs sampler, i.e., our method obtains the same estimation accuracy as the Gibbs sampler using less than one forty-sixth of the run length. If we combine this with the fact that the MJM method with $\theta = 0.0$ costs 2.021 times as much as the Gibbs sampler, then overall this method is still nearly 23 times more efficient than the Gibbs sampler.

In general, the MJM method performs better, i.e., it has faster convergence and better estimation performance, than the Gibbs sampler when applied to sampling from

the target distribution $\pi(\boldsymbol{\delta})$ in this example. We have also found that comparisons of the MJM method versus the Gibbs sampler show similar features for other choices of the priors. For example, when using $\alpha_0 = 0.2$ and $k = 5$ the MJM method is still more efficient than the Gibbs sampler as its mode jumping rate is more than 68 times higher and it is at least 150 times more efficient. If we use higher values of k, for example $k = 8$, the MJM method's mode jumping rate is 80 times higher than that for the Gibbs sampler and it is at least 88 times more efficient.

If we compare the results of sampling from $\pi(\boldsymbol{\delta})$ given in Table 6.4 with the results of sampling from $\pi(\boldsymbol{\beta}, \sigma^2, \alpha, \boldsymbol{\delta})$ given in Table 6.2, we can see that they are fairly similar for the MJM method except for the total amount of computation used as now one mode jumping step costs 2.021 cycles of the Gibbs sampler update instead of 3. A proper comparison of efficiency would also depend on the computational time needed to run a cycle of Gibbs sampler in both cases, which in turn depends on the exact implementation of the method and the programming code used. In each case there are various ways in which sampling can be made computationally faster. However, this was not our main concern at this stage and we shall not try to make a precise comparison of the efficiency of both methods.

The increase in mixing when sampling from $\pi(\boldsymbol{\delta})$ instead of $\pi(\boldsymbol{\beta}, \sigma^2, \alpha, \boldsymbol{\delta})$ is especially apparent in the case of the Gibbs sampler. When we compare the results for the Gibbs sampler given in Table 6.2 and Table 6.4, we can see that when sampling from $\pi(\boldsymbol{\delta})$ the mode jumping rate is more than twice as high as the rate when sampling from $\pi(\boldsymbol{\beta}, \sigma^2, \alpha, \boldsymbol{\delta})$, and the value of $\hat{\tau}^{(1)}$ for the first case is less than half of the value for the second case.

6.7 Applying the MJD method to sample the marginal posterior distribution of $\boldsymbol{\delta}$

6.7.1 General methodology

In the MJM method each mode jumping step costs at least twice as much as the Gibbs sampler. This is because each mode jumping step involves two cycles of the Gibbs sampler update, where each cycle of the Gibbs sampler is used to update the corresponding mode location. However, if instead of updating the whole mode location we only update the $z^{(i)}$ where there are differences between the mode locations, i.e., just doing a restricted Gibbs sampling, then each mode jumping step will cost less. Therefore we shall apply the MJD method which uses a restricted Gibbs sampler when proposing a mode jumping step to sample from the the target distribution $\pi(\boldsymbol{z})$. In this method, we shall go through the three stages: 1) initial exploration; 2) clustering;

and 3) sampling. While the initial exploration and clustering stages remain the same as the ones in the MJM method, as described in section 6.6.1, the sampling stage needs to be defined differently.

Sampling

After going through the initial exploration and clustering stages, we shall end up with m final mode locations, $\boldsymbol{\eta}_j$, $j = 1, \ldots, m$. The mode jumping step using differences then follows as in section 3.3.4. Note that here we shall again use $\boldsymbol{z}_q$ to denote the proposal state instead of $\boldsymbol{y}$, and correspondingly use $\boldsymbol{z}'_q$ instead of $\boldsymbol{y}'$. To apply this mode jumping step we define $\boldsymbol{d}_{j,\,k} = \boldsymbol{\eta}_k - \boldsymbol{\eta}_j$. We take the operator $\oplus$ to be a restricted addition so that $a = b \oplus c$ means $a^{(i)} = \max\left(\min(b^{(i)} + c^{(i)}, 1),\, 0\right)$ for $i = 1, \ldots, n$. We define a vector $\boldsymbol{R} = (R^{(1)}, \ldots, R^{(n)})$ where

$$R^{(i)} = \begin{cases} 1 & \text{if } \eta_j^{(i)} \neq \eta_k^{(i)} \text{ for at least one pair } j,\, k, \\ 0 & \text{otherwise.} \end{cases}$$

If $R^{(i)} = 1$, this means the value of the corresponding $Z^{(i)}$ may change in the step from $\boldsymbol{z}$ to $\boldsymbol{z}'_q = \boldsymbol{z} \oplus \boldsymbol{d}_{h(z),j}$, and therefore observation i is an outlier in some modes and not an outlier in others. This implies that the conditional distributions of the $Z^{(i)}$ where $R^{(i)} = 1$ are the most likely cases to be affected when the values of some elements of $\boldsymbol{Z}$ are changed. Therefore we set the random perturbation process to be an update using one cycle of a restricted Gibbs sampler on $\boldsymbol{z}'_q$ where we only update components i where $R^{(i)} = 1$. Correspondingly, in this case $q(\boldsymbol{z}'_q, \boldsymbol{z}_q)$ is the probability of moving from state $\boldsymbol{z}'_q$ to the proposal state $\boldsymbol{z}_q$ via one cycle of the restricted Gibbs sampler. Using a restricted Gibbs sampling will greatly reduce the computational cost of this method, especially if the size of the differences, i.e., the number of i where $R^{(i)} = 1$ is small compared to the length of $\boldsymbol{Z}$, n.

6.7.2 Application to Example 5

Suppose we want to identify the outliers in the star cluster CYG OB1 data set by sampling from the target distribution $\pi(\boldsymbol{\delta})$ defined by equation (6.5.1) using the MJD method.

Implementation

Implementation of the initial exploration, clustering and sampling stages are the same as in section 6.6.2, except for mode jumping step that we use in the sampling stage as we are going to use the MJD method instead, as described in section 6.7.1. Using the

two mode locations found, as given in Table 6.3, we find the vector $\boldsymbol{R}$ to have

$$R^{(i)} = \begin{cases} 1, & i = 11,\ 20,\ 30,\ 34, \\ 0, & \text{elsewhere.} \end{cases}$$

Results for Example 5

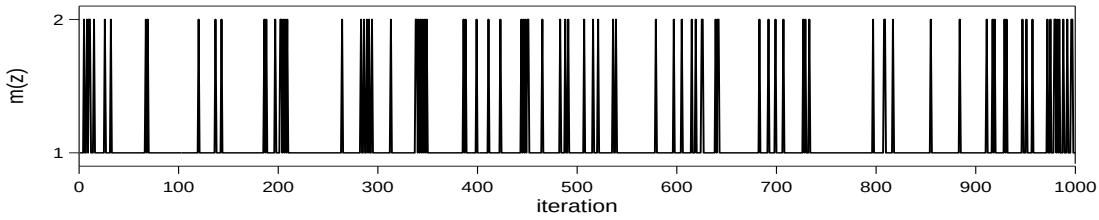
We simulated a chain of length 100,000 using the MJD method. We choose $\theta = 0.1$, 0.25, 0.5, and 0.75. Here we choose $\theta = 0.1$ rather than $\theta = 0$ because if we were to use $\theta = 0$ then elements $Z^{(i)}$ where $R^{(i)} = 0$ would never get updated. The values of $m(\boldsymbol{z})$ for the first 1,000 iterations of the various simulated chains are shown in Figure 6-6. From this figure we can see that the chains simulated using this method are mixing fairly well. The estimate from these 100,000 iterations of the transition matrix for the movement of the chain between the two modes for this method with $\theta = 0.25$ is given by

$$\widehat{P}'_d = \begin{pmatrix} 0.9108 & 0.0892 \\ 0.6664 & 0.3336 \end{pmatrix}.$$

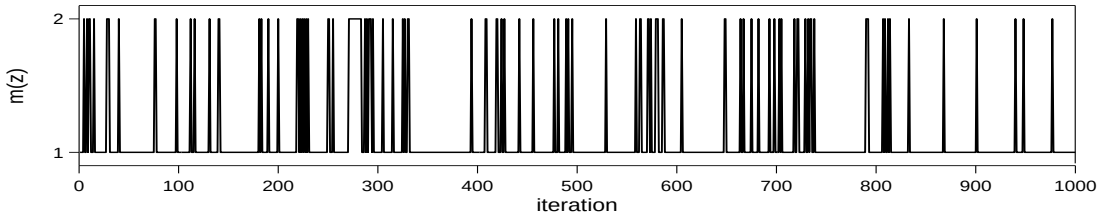
If we compare this with the estimate for the Gibbs sampler, $\widehat{P}'_{gs}$, and the estimate for the MJM method with $\theta = 0.25$, $\widehat{P}'_m$, we can see that the values off the diagonal in $\widehat{P}'_d$ are substantially larger than the values in $\widehat{P}'_{gs}$ but slightly smaller than the values in $\widehat{P}'_m$. This indicates that the MJD method is mixing better than the Gibbs sampler but not quite as well as the MJM method.

The overall results of the MJD method are summarized in Table 6.5. If we compare the results of sampling using the MJD method to those of the MJM method (as given in Table 6.4), then we can see that the mode jumping rates for the MJD method are 10% less than the MJM method, the values of λ^* are slightly higher and the values of $\hat{\tau}(f^{(i)})$ are around 20% greater than those for the MJM method.

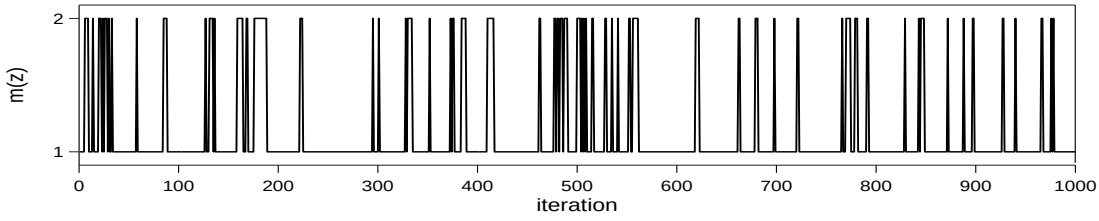
However, the MJD method costs considerably less than the MJM method. This is because the restricted Gibbs sampler will cost less than the usual Gibbs sampler as the size of the differences, i.e., the number of i where $R^{(i)} = 1$ is small compared to the data size n. In each iteration we choose to do Gibbs sampling with probability θ and to do a mode jumping step with probability $1 - \theta$, where for each mode jumping step we are using two restricted cycles of the Gibbs sampler update and one evaluation of $\pi(\boldsymbol{z}_q)/\pi(\boldsymbol{z})$. This means that the MJD method uses θ Gibbs sampler updates, $2(1-\theta)$ restricted Gibbs sampler updates and $(1-\theta)$ evaluations of $\pi(\boldsymbol{z}_q)/\pi(\boldsymbol{z})$. In this example, the number of i where $R^{(i)} = 1$ is only 4, which means that the restricted cycle of the Gibbs sampler costs only 8.5% of the usual Gibbs sampler; the MJD method then uses $(0.17 + 0.83\theta)$ Gibbs sampler updates and $(1 - \theta)$ evaluations of $\pi(\boldsymbol{z}_p)/\pi(\boldsymbol{z})$. When we look at the total amount of computation used by both methods as given in Table



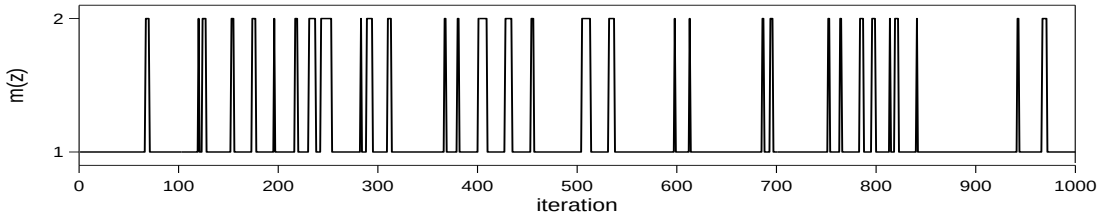
(a) $\theta = 0.10$



(b) $\theta = 0.25$



(c)$\theta = 0.50$



(d) $\theta = 0.75$

Figure 6-6: Plots of the values of $m(\boldsymbol{z})$ for the first 1,000 iterations, where the chains are simulated using the MJD method with various values of θ.

	Method				
	MJD				Gibbs
	$\theta = 0.1$	$\theta = 0.25$	$\theta = 0.5$	$\theta = 0.75$	sampler
Mode jumping rate (%)	18.77	15.73	10.89	5.95	1.03
Total computation ($\times 10^5$)					
a) initial exploration					
i) Gibbs cycles	0.2	0.2	0.2	0.2	0
b) sampling					
i) Gibbs cycles	0.253	0.378	0.585	0.793	1
ii) evaluations of $\pi(\boldsymbol{z}_q)/\pi(\boldsymbol{z})$	0.9	0.75	0.5	0.25	0
iii) total	0.272	0.394	0.596	0.798	1
Eigenvalue λ_1 of $\widehat{P}$	0.095	0.244	0.480	0.716	0.946
Estimated IAC for $f^{(1)} = I\{m(\boldsymbol{z}) = 1\}$ calculated using $\widehat{P}$					
$\hat{\tau}(f^{(i)})$	1.21	1.65	2.84	6.04	35.98

Table 6.5: Overall results for a chain of run length 100,000, simulated using the MJD method and the Gibbs sampler.

6.5 and the previous Table 6.4, then we can see that sampling using the MJD method costs considerably less than sampling using the MJM method. For example, in the case where $\theta = 0.25$ the MJD method costs only 22.3% of the MJM method. Then in this case even though the value of $\hat{\tau}(f^{(i)})$ for the MJD method is around 1.21 times bigger than the MJM method, it is still 3.7 times more efficient. This means that the MJD method is more efficient than the MJM method.

If we compare the results of the MJD method with $\theta = 0.1$ with those of the Gibbs sampler, the MJD method has a higher mode jumping rate (more than 18 times higher), costs only 27.2% as much, and obtains the same estimation accuracy as the Gibbs sampler using less than one twenty-ninth of the run length, which means that in this case the MJD method is around 109 times more efficient than the Gibbs sampler. This means that in general, the MJD method performs the best among all three methods.

Note that the MJD method applied to this problem is similar to the one applied to the image analysis problem in chapter 5, where in both cases the target distributions have binary variables. The MJD method in this problem involves flipping a particular set of values connected to moving between two particular modes. We also obtain a saving in computation as there is no need to update the whole parameter vector after flipping the values, just a subset of it. Flipping a set of values here is similar to "features" that appear or disappear in the image analysis problem. However, in the image analysis problem "features" involve the concept of "neighbourhood", i.e., a set of neighbouring pixels; in this problem we have no definite concept of "neighbourhood"

for $\boldsymbol{\delta}$. $\boldsymbol{R}$ would then represent a kind of "neighbourhood" for $\boldsymbol{\delta}$, as the conditional distributions of the $\delta^{(i)}$ where $R^{(i)} = 1$ are the most likely cases to be affected when the values of some elements of $\boldsymbol{\delta}$ are changed.

6.8 Finding additional modes in the marginal posterior distribution of $\boldsymbol{\delta}$

6.8.1 General methodology

One issue that can arise when applying the mode jumping approach to sample from the target distribution $\pi(\boldsymbol{\delta})$ is the possibility of missing a mode during the initial exploration stage. One way to ensure that we have the locations of all the possible modes is to consider all 2^n possible combinations of 0 and 1. However, the total number of these combinations increases with the value of n, and if n is large there would be too many combinations to consider.

To reduce the number of combinations, we can just consider a set of combinations of the mode locations that we have already found in our initial exploration. We do this by focusing only on values that change in our list of mode locations. We go through the clustering stage where we cluster the mode locations that we found in the initial exploration stage to get m mode locations which we denote as $\boldsymbol{\eta}_j$, $j = 1, \ldots, m$, and find the corresponding vector $\boldsymbol{R} = (R^{(1)}, \ldots, R^{(n)})$ defined in section 6.7.1. We then fix those $\eta^{(i)}$ where $R^{(i)} = 0$ at a value common to all mode locations $\boldsymbol{\eta}_j$ and combine these with all possible combinations of 0 and 1 for the $\eta^{(i)}$ where $R^{(i)} = 1$. If the total number of i where $R^{(i)} = 1$ is q, then we have 2^q potential local modes.

However, we can further reduce the number of possible combinations by clustering the values of i where $R^{(i)} = 1$, and then looking at the possible combinations using these clusters. We shall illustrate this by using an example. Consider three vectors $\boldsymbol{\eta}_1 = (0, 1, 1, 1, 1)$, $\boldsymbol{\eta}_2 = (0, 0, 0, 1, 1)$ and $\boldsymbol{\eta}_3 = (0, 0, 0, 0, 0)$, with their corresponding $\boldsymbol{R} = (0, 1, 1, 1, 1)$. Therefore we are interested in $i = 2, 3, 4, 5$. If any of these components have the same value for all $\boldsymbol{\eta}_j$, they are put into a set and we generate a corresponding vector $\boldsymbol{W}$ where $W^{(i)} = 1$ if i is in this set and $W^{(i)} = 0$ otherwise. For this example $i = 2$ and 3 have the same value for all $\boldsymbol{\eta}_j$, and so do $i = 4$ and 5. Therefore we obtain $\boldsymbol{W}_1 = (0, 1, 1, 0, 0)$ and $\boldsymbol{W}_2 = (0, 0, 0, 1, 1)$. We shall collect all these $\boldsymbol{W}$, say m' in total, then use these $\boldsymbol{W}_k$ in a cross-over operation, which is similar to the one we use in the image analysis problem in the previous chapter. In this cross-over operation, we shall consider all possible combinations of the mode locations with the vectors $\boldsymbol{W}_k$, $k = 1, \ldots, m'$, where each $\boldsymbol{W}_k$ could either be absent or present, to get $2^{m'}$ modes. For this example we obtain 4 mode locations, which include the

original three vectors and a new one: $(0, 1, 1, 0, 0)$.

After obtaining all these new mode locations, we apply a restricted hill-climbing process to these modes, where we only do hill-climbing on the values of $\eta_j^{(i)}$ where $R^{(i)} = 1$, to ensure that they are real mode locations. Then we remove any repeating mode location if any exists. In the end, we might end up with more modes, or we could get back the original m modes. However, at least this cross-over operation provides us with a way to find additional modes if any exist.

6.8.2 Example 6: Stack loss data

We look at another real data set: the "stack loss data", which is a group of data from a plant for the oxidation of ammonia to nitric acid. In this data set, $n = 21$ diary observations were collected for three explanatory variables and one response variable. The data may be found in Daniel & Wood (1980), and are given in Appendix D. We assume that the data were generated by the normal scale contamination model. Then for this example the posterior distribution for the outlier labels, $\pi(\boldsymbol{\delta})$, is given by equation (6.5.1). For illustration purposes, we choose $\gamma_1 = 3$, $\gamma_2 = 18$ and $k = 7$. Note that this data set was discussed by Justel & Peña (1996) where they showed that the Gibbs sampler identified observations 1, 3, 4 and 21 as outliers.

6.8.3 Application of the MJD method to Example 6

We apply the MJD method, as described in section 6.7.1, to sample from the target distribution $\pi(\boldsymbol{\delta})$ defined by equation (6.5.1).

Implementation

1. Initial exploration
   We carried out a run of length 200 using the Gibbs sampler with simulated annealing, using the logarithmic temperature function, $T(k) = 1/\ln(1 + k)$, and applied the hill-climbing process to the end values. We repeated the procedure of simulated annealing and hill-climbing 100 times. For 50 runs, we used starting values $\boldsymbol{z} = \boldsymbol{\delta}_s$ with $\sum_i (1 - \delta_s^{(i)}) = 3$, where the three i with $\delta_s^{(i)} = 0$ are uniformly chosen. For the other 50 we used the starting values $\boldsymbol{\delta}_s$ where $\sum_i (1 - \delta_s^{(i)}) = 18$ and the three i with $\delta_s^{(i)} = 1$ are uniformly chosen. These runs then produced 100 mode locations, only three of which were distinct.

2. Clustering
   For the clustering algorithm, we chose $\xi = 1.0$ and obtained $m = 3$. The values of the corresponding mode locations $\boldsymbol{\eta}_i$, $i = 1, 2, 3$, are given in Table 6.6. From

Mode location	Observation i where $\eta^{(i)} = 1$
$\boldsymbol{\eta}_1$	–
$\boldsymbol{\eta}_2$	4, 21
$\boldsymbol{\eta}_3$	1, 3, 4, 21

Table 6.6: Values of the mode locations for the stack loss data.

Mode location	Observation i where $\eta^{(i)} = 1$
$\boldsymbol{\eta}_1$	–
$\boldsymbol{\eta}_2$	4, 21
$\boldsymbol{\eta}_3$	1, 3, 4, 21
$\boldsymbol{\eta}_4$	1, 3

Table 6.7: Values of the mode locations for the stack loss data after the cross-over operation.

these three mode locations we obtained the vector $\boldsymbol{R}$ where

$$R^{(i)} = \begin{cases} 1, & i = 1, 3, 4, 21, \\ 0, & \text{elsewhere} \end{cases}$$

and correspondingly the vectors $\boldsymbol{W}_k$, $k = 1, 2$, where

$$W_1^{(i)} = \begin{cases} 1, & i = 1, 3, \\ 0, & \text{elsewhere} \end{cases}$$

and

$$W_2^{(i)} = \begin{cases} 1, & i = 4, 21, \\ 0, & \text{elsewhere.} \end{cases}$$

Using these $\boldsymbol{W}_k$ in the cross-over operation we obtained $2^2 = 4$ mode locations which are given in Table 6.7. Among these four mode locations, three, i.e., $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_3$, were already found in the initial exploration stage while one, i.e., $\boldsymbol{\eta}_4$, is new. However, after applying the hill-climbing process to these four mode location, we found that $\boldsymbol{\eta}_4$ is not a real mode location as its values actually changed to the values of $\boldsymbol{\eta}_1$ during the hill-climbing process. Because $\boldsymbol{\eta}_4$ had become a duplicate of $\boldsymbol{\eta}_1$, it was removed and we ended up with the original three mode locations.

3. Sampling
   We combine local movements using the Gibbs sampler and mode jumping steps: at each iteration we choose to do Gibbs sampling with probability θ and to do a mode jumping step with probability $1 - \theta$. The mode jumping step proceeds as given in section 6.6.1, where we choose to jump to a different mode with equal probability, i.e., $p_{ij} = 1/2$ for $j \in \{1, \ldots, 3\}$, $j \neq i$.

	Method				
	MJD				Gibbs
	$\theta = 0.1$	$\theta = 0.25$	$\theta = 0.5$	$\theta = 0.75$	
Mode jumping rate (%)	32.36	30.00	26.47	23.26	19.24
Total computation ($\times 10^5$)					
a) initial exploration					
i) Gibbs cycles	0.2	0.2	0.2	0.2	0
b) sampling					
i) Gibbs cycles	0.442	0.535	0.69	0.845	1
ii) evaluations of $\pi(\boldsymbol{z}_q)/\pi(\boldsymbol{z})$	0.9	0.75	0.5	0.25	0
iii) total	0.485	0.571	0.714	0.857	1
Eigenvalue λ_k of $\widehat{P}$ for which $\lvert\lambda_k\rvert = \lambda^*$	0.441	0.501	0.596	0.686	0.783
Estimated IAC for $f^{(1)} = I\{m(\boldsymbol{z}) = 1\}$ calculated using $\widehat{P}$					
i) $\hat{\tau}(f^{(1)})$	1.73	2.03	2.61	3.53	5.36
ii) $\hat{\tau}(f^{(2)})$	1.86	2.01	2.34	2.73	3.47
iii) $\hat{\tau}(f^{(3)})$	2.58	3.00	3.91	5.27	7.95

Table 6.8: Overall results for a chain of run length 100,000, simulated using the MJD method and the Gibbs sampler.

Results for Example 6

When comparing sampling from the target distribution $\pi(\boldsymbol{z})$ using the MJD method and the Gibbs sampler, we define the nearest mode function $m(\boldsymbol{z})$ so that $m(\boldsymbol{z}) = 3$ if $\delta^{(i)} = 1$ for $i = 1, 3, 4, 21$, $m(\boldsymbol{z}) = 2$ if $\delta^{(i)} = 1$ for $i = 4, 21$ but not for both $i = 1$ and 3, and $m(\boldsymbol{z}) = 1$ for any other cases.

We simulated a chain of length 100,000 using the MJD method and the Gibbs sampler. For our mode jumping method, we choose $\theta = 0.1$, 0.25, 0.5, and 0.75. The overall results of both methods are summarized in Table 6.8. From Table 6.8, we can see that the MJD method yields higher mode jumping rates than the Gibbs sampler, costs less, and has smaller values of λ^* and $\hat{\tau}(f^{(i)})$. In fact, when we include the cost of sampling, the MJD method with $\theta = 0.1$ is 6 times more efficient than the Gibbs sampler. In general, we can see that the MJD method also performs better than the Gibbs sampler for this example.

6.9 The effect of the priors on the normal contamination model

Box & Tiao (1968) suggest that Bayesian linear regression using the normal scale contamination model would be insensitive to the values of α and k in the ranges

	α_0		
k	0.05	0.1	0.2
3	0.006	0.03	0.16
4	0.010	0.07	0.36
5	0.016	0.12	0.52
6	0.020	0.14	0.62
7	0.021	0.15	0.65
8	0.020	0.15	0.65
9	0.016	0.13	0.63
10	0.012	0.11	0.59

Table 6.9: The estimates of the probability associated with mode 2, i.e., the mode with outliers, for the star cluster data in Example 5 when sampled using different values of α_0 and k.

$0.01 \leq \alpha \leq 0.1$ and $3 \leq k \leq 10$. We have used the MJD method to investigate the effect the values of α_0 and k have on the results from fitting the normal contamination model. For example, estimates of the probability associated with mode 2, i.e., the mode with outliers, for the star cluster data in Example 5 when sampled using different values of α_0 and k are shown in Table 6.9. From this table we can see that a different value for α_0 would result in a different probability associated with mode 2, a higher value of α_0 resulting in a bigger probability for this mode. In fact, for $\alpha_0 \leq 0.1$ the probability associated with the mode with outliers seems to be around 0.15 or less. This means that the model favours the mode with no outliers considerably, which does not appear to be a good analysis for this data set. When $\alpha_0 = 0.2$, the probability associated with mode 2 is approximately 0.6, therefore the model favours the mode with outliers more than the mode with no outliers. From this table we can also see that lower values of k, where $k < 5$, would result in a smaller probability associated with mode 2.

For the stack loss data in Example 6, estimates of the probability associated with mode 3, i.e., the mode with 4 outliers, when sampled using different values of α_0 and k are shown in Table 6.10. From this table we can see that a different value for α_0 would also result in a different probability associated with mode 3. Here a higher value of α_0 would again result in a bigger probability for this mode and lower values of k would result in a smaller probability.

Overall, these results seem to imply that the normal contamination model is in fact sensitive to the prior that we choose for α and the value that we choose for k.

	α_0		
k	0.05	0.1	0.2
3	0.05	0.14	0.34
4	0.12	0.29	0.56
5	0.17	0.40	0.70
6	0.19	0.46	0.76
7	0.20	0.48	0.80
8	0.18	0.48	0.81
9	0.16	0.44	0.80
10	0.14	0.41	0.79

Table 6.10: The estimates of the probability associated with mode 3, i.e., the mode with 4 outliers, for the stack loss data in Example 6 when sampled using different values of α_0 and k.

6.10 Summary and discussion

We have shown how the mode jumping approach can be applied to Bayesian linear regression, in particular to the problem of outlier detection. Using a few examples, we have shown that the mode jumping approach performs well, and it performs much better than the usual Gibbs sampler as it gives better mixing and is more efficient. One of the major advantages of the mode jumping approach is that through our initial exploration, we know which different modes exist. We are also able to estimate the probability of subsets of data outlying, i.e., the probability associated with each mode.

Justel & Peña (1996) state that "when the set of data has many outliers that mask each other, Gibbs sampling will fail and posterior distributions are poorly estimated". However, a question arises whether the low posterior probability of each of the four observations being labelled as outliers is caused by the Gibbs sampler failing to detect these outliers, or by the normal contamination model itself giving this set of observations a small probability of being labelled as outliers. These two issues could be mistaken to be one single issue, and it is hard to resolve this question since the Gibbs sampler does not mix well between the sets of masking outliers. However, we have an advantage since our mode jumping method *does* mix well between the modes, therefore we can estimate the probability associated with this set of observations being labelled as outliers. Our results indicate that the probability of a set of observations being labelled as outliers under the normal contamination model is influenced significantly by the prior that we choose for α and the value that we choose for k. This would imply that the normal scale contamination model has limitations as a method for detecting masking outliers in Bayesian linear regression.

Mohr (2007) claims that the normal scale contamination model is better suited to scattered outliers rather than clustered outliers. An alternative model that has been suggested is a mixture of regression models. For example, Hurn et al. (2003) presented a Bayesian method for switching regression models, that allow for more than two groups and for non-normal distributions of the data. Mohr (2007) on the other hand introduced a Bayesian model which represents outliers that cluster, in the form of a mixture of some special regression models. Since these methods need MCMC samplers that can deal with multiple modes, therefore our methods could be applied here too.

Chapter 7

Application IV: Variable dimension distributions

In this chapter, we shall consider sampling problems where the target distribution has variable dimension, i.e., the variables have state subspaces of differing dimensionality. We shall apply our mode jumping approach to this kind of problems and assess its performance.

7.1 Reversible jump MCMC methodology

Distributions with state subspaces of differing dimensionality arise in problems where the number of parameters is not fixed. Some examples of this kind of problems are variable selection in regression, Bayesian choice between models with different numbers of parameters, mixture deconvolution with an unknown number of components and image segmentation. Green (1995) suggested a framework where MCMC can be used to sample from this kind of distributions, and the result is called *reversible jump MCMC* or RJMCMC. In this section, we shall introduce RJMCMC in a general setting, summarizing the method as described by Green (1995) and Green (2003).

Suppose our target distribution is a sum of densities over dimensions 1 to N. Let $\pi(k, x)$ denote the sub-density in dimension $K = k$, and let $S_k \subseteq \mathbb{R}^k$ denote the state subspace of X in this dimension. Then the target distribution is a sum of N sub-densities:

$$\begin{cases} \pi(1, x), & \text{a sub-density for } x \in S_1 \\ \vdots & \\ \pi(N, x), & \text{a sub-density for } x \in S_N. \end{cases}$$

We shall refer to this target distribution as "π". Let

$$p_1 = \int_{S_1} \pi(1, x)\,\mathrm{d}x, \ \ldots\ , p_N = \int_{S_N} \pi(N, x)\,\mathrm{d}x.$$

Then we can also interpret the target distribution π as a mixture of N components:

$$\text{with probability } p_k, \quad X \sim \frac{1}{p_k}\pi(k,x), \qquad x \in S_k, \quad k = 1, \ldots, N,$$

where each $\pi(k,x)/p_k$ is now a proper density in dimension k.

We denote the state of the RJMCMC sampler at time t by (K_t, X_t), where K_t is the dimension at time t and X_t is the K_t-dimensional variable. Let the current state be $(K_t, X_t) = (k, x)$. The RJMCMC sampler is made up of a variety of move types, ϕ, each of which moves between two of the dimensions 1 to N. Formally, we denote this pair of dimensions for move type ϕ by $\{k_1(\phi), k_2(\phi)\}$. Some move types may operate within a fixed dimension, in which case $k_1 = k_2$. When in state (k, x) we choose a move type ϕ with probability $q_\phi(k, x)$, restricting choice to ϕs for which $k_1(\phi) = k$ or $k_2(\phi) = k$ (or both). Let k' be $k_2(\phi)$ if $k_1(\phi) = k$ or $k_1(\phi)$ if $k_2(\phi) = k$, then a move of type ϕ will propose a new state (k', x') where $x' \in S_{k'}$. The variable x' is generated from a density $f_\phi((k, x), (k', x'))$ on $S_{k'}$ and this is accepted with probability

$$\alpha\left((k,x),(k',x')\right) = \min\left\{1, \frac{\pi(k',x')\, q_\phi(k',x')\, f_\phi\left((k',x'),(k,x)\right)}{\pi(k,x)\, q_\phi(k,x)\, f_\phi\left((k,x),(k',x')\right)}\right\}.$$

This process defines a transition kernel P_ϕ which we now show satisfies detailed balance within move type ϕ. Suppose $A \subset S_k$ and $B \subset S_{k'}$. For detailed balance we need

$$\begin{aligned}
&\mathbb{P}(K_t = k \,\text{and}\, X_t \in A \quad \text{then} \quad K_{t+1} = k' \,\text{and}\, X_{t+1} \in B) \\
&= \mathbb{P}(K_t = k' \,\text{and}\, X_t \in B \quad \text{then} \quad K_{t+1} = k \,\text{and}\, X_{t+1} \in A)
\end{aligned}$$

when (K_t, X_t) are sampled under the target distribution and we apply one step of RJMCMC. That is, we need

$$\begin{aligned}
&\int_{x\in A}\int_{x'\in B} \pi(k,x)\, P_\phi\left((k,x),(k',x')\right)\, \mathrm{d}x'\, \mathrm{d}x \\
&= \int_{x'\in B}\int_{x\in A} \pi(k',x')\, P_\phi\left((k',x'),(k,x)\right)\, \mathrm{d}x\, \mathrm{d}x'. \qquad (7.1.1)
\end{aligned}$$

For any pair of states $x \in A$ and $y \in B$ with $y \neq x$,

$$\begin{aligned}
&\pi(k,x)\, P_\phi\left((k,x),(k',x')\right) \\
&= \pi(k,x)\, q_\phi(k,x)\, f_\phi((k,x),(k',x'))\, \alpha\left((k,x),(k',x')\right) \\
&= \pi(k,x)\, q_\phi(k,x)\, f_\phi\left((k,x),(k',x')\right)\, \min\left\{1, \frac{\pi(k',x')\, q_\phi(k',x')\, f_\phi\left((k',x'),(k,x)\right)}{\pi(k,x)\, q_\phi(k,x)\, f_\phi\left((k,x),(k',x')\right)}\right\}.
\end{aligned}$$

Considering the two cases where

$$\pi(k',x')\, q_\phi(k',x')\, f_\phi\left((k',x'),(k,x)\right) > \pi(k,x)\, q_\phi(k,x)\, f_\phi\left((k,x),(k',x')\right)$$

and vice versa, we see that

$$\begin{aligned}
&\pi(k,x)\, q_\phi(k,x)\, f_\phi\left((k,x),(k',x')\right)\, \min\left\{1, \frac{\pi(k',x')\, q_\phi(k',x')\, f_\phi\left((k',x'),(k,x)\right)}{\pi(k,x)\, q_\phi(k,x)\, f_\phi\left((k,x),(k',x')\right)}\right\} \\
&= \pi(k',x')\, q_\phi(k',x')\, f_\phi\left((k',x'),(k,x)\right)\, \min\left\{1, \frac{\pi(k,x)\, q_\phi(k,x)\, f_\phi\left((k,x),(k',x')\right)}{\pi(k',x')\, q_\phi(k',x')\, f_\phi\left((k',x'),(k,x)\right)}\right\}
\end{aligned}$$

and, therefore,

$$\begin{aligned}
&\pi(k,x)\, P_\phi\left((k,x),(k',x')\right) \\
&= \pi(k',x')\, q_\phi(k',x')\, f_\phi\left((k',x'),(k,x)\right)\, \min\left\{1, \frac{\pi(k,x)\, q_\phi(k,x)\, f_\phi\left((k,x),(k',x')\right)}{\pi(k',x')\, q_\phi(k',x')\, f_\phi\left((k',x'),(k,x)\right)}\right\} \\
&= \pi(k',x')\, q_\phi(k',x')\, f_\phi\left((k',x'),(k,x)\right)\, \alpha\left((k',x'),(k,x)\right) \\
&= \pi(k',x')\, P_\phi\left((k',x'),(k,x)\right).
\end{aligned}$$

For any pair of states $x \in A$ and $y \in B$ with $y = x$ it is true trivially that $\pi(k,x)\, P_\phi\left((k,x),(k',x')\right) = \pi(k',x')\, P_\phi\left((k',x'),(k,x)\right)$. Hence, equation (7.1.1) holds since the two integrands are equal.

If the transition kernel for each move type ϕ satisfies detailed balance, then the overall transition kernel satisfies general balance (for the detailed proof, see section 2.1.4). Note that for our purposes we are interested in the case where $f_\phi\left((k,x),(k',x')\right)$ is a proper density on $\mathbb{R}^{k'}$ with full dimensionality k'. There are other versions of the RJMCMC method where $f_\phi\left((k,x),(k',x')\right)$ would only be supported on values of x' in a lower sub-dimensional subset of $\mathbb{R}^{k'}$.

In practice, dimension jumping moves often have low acceptance rates. This is because a random proposal may be unlikely to "hit" the high density area of π in a new dimension, even without there being several modes per dimension. Special types of moves can be defined to take advantage of the structure of the variables. For example, in modelling a density by a mixture of normals, Green & Richardson (1997) use moves which split a normal kernel or merge two kernels. Similarly, Al-Awadhi et al. (2004) define birth, death, merge and split moves in an image analysis application where the image is made up of unknown number of elliptical objects. These moves are meant to help obtain a good proposal when jumping to a new dimension. The approach we introduce in the next section will look simple relative to these other methods. This is because our approach spends effort gathering information about the distribution π and its modes in a substantial initial exploration, after which it is relatively straightforward to propose good moves between dimensions.

7.2 General methodology of the mode jumping approach

Application of our mode jumping approach to sample from a target distribution of variable dimension is similar to the fixed dimension case, as described in section 3.3. Then to apply our method, we shall go through the four stages:

1. Initial exploration, where we try to find the locations of all the modes.
2. Clustering, where we cluster the mode locations and find a single mode location to represent each cluster.
3. Modelling, where we fit a approximate model at each of the mode locations obtained from the clustering stage.
4. Sampling, where we use the models from the modelling stage to jump between the modes.

Note that for the target distribution π, we assume we know the range of values for K, 1 to N say, and we know $\pi(k, x)$ up to an overall normalizing constant, i.e., we know $\pi(k, x) = c\,\psi(k, x)$ for all k and x, where $\psi(k, x)$ is known but c is unknown.

7.2.1 Initial exploration

The aim of the initial exploration is to explore the target distribution's whole support and to find all possible mode locations. In this case, we shall run the initial exploration in each dimension k separately, using optimization methods to search for the mode locations. Possible optimization methods that can be used at this stage have been discussed in section 3.3.1. If we repeat the runs l times in each dimension $k \in \{1, \ldots, N\}$ using a set of random starting points, at the end of this stage we shall have $l \times N$ mode locations. By design, there will be at least one mode in each dimension.

7.2.2 Clustering

After finding $l \times N$ mode locations in the initial exploration, we want to cut down the number of modes by using clustering. We shall do the clustering in each dimension separately, grouping together mode locations which are close to each other. In each dimension k, after we use a clustering algorithm to cluster the l mode locations into, say, m_k clusters, we choose one mode location to represent each cluster. Typically, we choose the mode location with the highest probability density, i.e., the highest value of $\pi(k, x)$ within the cluster. At the end of the clustering stage, we end up with $m = \sum_{k=1}^{N} m_k$ mode locations, which we denote by (k_i, η_i), $i = 1, \ldots, m$. Again, by design, this set of modes will include at least one in each dimension.

7.2.3 Modelling

At each mode location (k_i, η_i), we fit an approximate local distribution $g_i(k_i, x)$ and possibly a weight w_i. We shall also consider versions of our method in which this explicit modelling is not necessary and alternative methods are used to generate a proposal at a selected mode.

7.2.4 Sampling

To ensure that the sampling explores both within and between modes, we are going to combine two types of update: 1) local changes; and 2) a mode jumping step. In the fixed dimension case, the methods for the mode jumping step were divided into two types: 1) mode jumping using *modelling* (MJM); and 2) mode jumping using *differences* (MJD). In the variable dimension case we have only considered the MJM method as it is the more obvious choice to us and straightforward to implement. However, there could be an equivalent MJD method for the variable dimension case, where this method allows you to keep some parts of the X the same while just changing a subset of components: we shall leave this method as a topic for future research.

To apply our mode jumping methods, we shall define a measure of distance which allows us to compute a "nearest mode" function, $h(k, x) \in \{1, \ldots, m\}$, which specifies the nearest of the mode locations (k_i, η_i) to the state x. We also need to specify the probability of proposing a jump to mode j when currently at mode i, i.e., $p_{i,\,j}$, for i, $j \in \{1, \ldots, m\}$, where $j \neq i$. If we can estimate the weight w_i at each mode location η_i, we can use this to specify

$$p_{i,\,j} = \frac{w_j}{\sum_{k \neq i} w_k}, \qquad \text{for } j \neq i.$$

If estimated weights are not available, we choose to jump to a different mode with equal probability, i.e., $p_{i,j} = 1/(m-1)$ for $j \neq i$.

Mode jumping using modelling

When using the MJM method, there are two kinds of modelling that we can use: explicit modelling or implicit modelling.

1. *Explicit modelling* (MJM1)

In this method, we use the approximate local distributions $g_i(k_i, x)$, $i = 1, \ldots, m$, from the modelling stage to sample from the target distribution. At the start of this method, we assume that the probabilities $p_{i,\,j}$, i, $j = 1, \ldots, m$ and $j \neq i$, have already been defined. Then for one mode jumping step in this method, we:

1. Determine the nearest mode $h(k,x)$ to the current state (k,x).

2. Choose a different mode $j \in \{1,\ldots,m\}\backslash h(k,x)$ with probability $p_{h(k,x),j}$. We sample x' from $g_j(k_j,x')$, and set the proposal state to be $(k',x') = (k_j,x')$.

3. Determine the nearest mode $h(k',x')$ to (k',x').

   (a) If $h(k',x') \neq j$, we reject the proposal (k',x') and stay at (k,x).

   (b) If $h(k',x') = j$, we accept this proposal with probability

$$\alpha\left((k,x),(k',x')\right) = \min\left\{1, \frac{\pi(k',x')\, p_{h(k',x'),\,h(k,x)}\, g_{h(k,x)}(k,x)}{\pi(k,x)\, p_{h(k,x),\,h(k',x')}\, g_{h(k',x')}(k',x')}\right\}.$$

We can see that this mode jumping step satisfies the detailed balance equation (7.1.1) for any $A \subset S_k$ and any $B \subset S_{k'}$, where the move type ϕ refers to a jump between modes $h(k,x)$ and $h(k',x')$, $p_{h(k,x),\,h(k',x')}$ takes the role of $q_\phi(k,x)$ and $g_{h(k',x')}(k',x')$ is equivalent to $f_\phi\left((k,x),(k',x')\right)$. Note that this implies the sampling of $x' \in S_{k'}$ near mode $h(k',x')$ does not depend on the current value of x, apart from the fact that it is near mode $h(k,x)$.

2. *Implicit modelling* (MJM2)

In this method, we produce a randomly sampled value at a mode i by performing a cycle of updates on η_i using the Gibbs sampler with k_i fixed. At the start of this method, we assume that we know (k_i,η_i), $i = 1,\ldots,m$, from the initial exploration and clustering process, and that the probabilities $p_{i,\,j}$, $i,\ j = 1,\ldots,m$ and $j \neq i$, have already been defined. We shall define $r_i(k_i,x)$ to be the probability density of η_i changing to the state x during one cycle of the Gibbs sampler in dimension k_i. Then for one mode jumping step in this method, we:

1. Determine the nearest mode $h(k,x)$ to the current state (k,x).

2. Choose a different mode $j \in \{1,\ldots,m\}\backslash h(k,x)$ with probability $p_{h(k,x),j}$. We perform one cycle of the Gibbs sampler, starting from (k_j,η_j) to get (k_j,x'). The proposal is then $(k',x') = (k_j,x')$, with the corresponding probability density $r_j(k',x')$.

3. Determine the nearest mode $h(k',x')$ to (k',x').

   (a) If $h(k',x') \neq j$, we reject the proposal (k',x') and stay at (k,x).

   (b) If $h(k',x') = j$, we accept this proposal with probability

$$\alpha\left((k,x),(k',x')\right) = \min\left\{1, \frac{\pi(k',x')\, p_{h(k',x'),\,h(k,x)}\, r_{h(k,x)}(k,x)}{\pi(k,x)\, p_{h(k,x),\,h(k',x')}\, r_{h(k',x')}(k',x')}\right\}.$$

This mode jumping step satisfies detailed balance following the same argument as in the explicit modelling case but with the function g_i replaced by r_i.

7.3 Example 7: Mixture of Gaussian distributions in 2 dimensions

In this example we take the target distribution π to be a mixture of normal distributions, two on $\mathbb{R}$ and three on $\mathbb{R}^2$. We denote the probability density of a mixture of normal distributions on $\mathbb{R}$ by

$$f_1(x) = \sum_{i=1}^{2} \omega_i \, \mathrm{N}((k_i, \mu_i), \Sigma_i)(x),$$

where $\omega_1 = 0.3$ and $\omega_2 = 0.7$, $\mu_1 = (5)$, $\mu_2 = (-3)$ and $\Sigma_1 = (0.5)$, $\Sigma_2 = (0.25)$. Similarly, we denote the probability density of a mixture of normal distributions on $\mathbb{R}^2$ by

$$f_2(x) = \sum_{i=3}^{5} \omega_i \, \mathrm{N}_2((k_i, \mu_i), \Sigma_i)(x),$$

where $\omega_3 = 0.2$, $\omega_4 = 0.5$ and $\omega_5 = 0.3$, $\mu_3 = (0, 0)$, $\mu_4 = (7, 0)$, $\mu_5 = (0, -8)$ and

$$\Sigma_3 = \begin{pmatrix} 0.5 & 0.35 \\ 0.35 & 0.5 \end{pmatrix}, \quad \Sigma_4 = \begin{pmatrix} 0.25 & -0.15 \\ -0.15 & 0.25 \end{pmatrix}, \quad \Sigma_5 = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}.$$

We define π by defining its sub-densities in dimension 1 and 2 as

$$\begin{cases} \pi(1, x) = 3c \, f_1(x), & \text{for } x \in \mathbb{R}, \\ \pi(2, x) = 7c \, f_2(x), & \text{for } x \in \mathbb{R}^2, \end{cases} \tag{7.3.1}$$

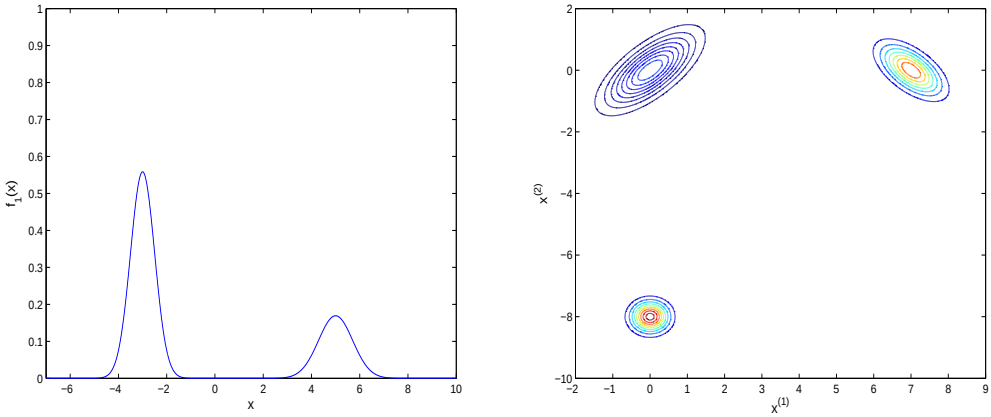
where c is the appropriate value to make this a probability distribution. With these choices π has 5 well separated modes, as shown in Figure 7-1. Since f_1 and f_2 are both proper densities, it is simple to deduce that $c = 0.1$ but we shall proceed as if c is unknown.

7.4 Application of the mode jumping approach to Example 7

We shall apply the MJM1 method to sample from the target distribution π defined by expression (7.3.1) in section 7.3.

7.4.1 Implementation of the mode jumping approach

1. Initial exploration



(a) $x \in \mathbb{R}$ (b) $x \in \mathbb{R}^2$

Figure 7-1: The distribution π in $\mathbb{R}$ and $\mathbb{R}^2$.

In each dimension $k = 1, 2$, we carried out a run of length 100 using the Metropolis-Hastings algorithm with simulated annealing, where at the j^{th} iteration we sampled from the target distribution over x of $\pi_{T(j)}(k, x) \propto \{\pi(k, x)\}^{1/T(j)}$. Here the logarithmic temperature function, $T(j) = 1/\ln(1 + j)$, is used. For $k = 1$, we used the proposal distribution

$$q_1(x, y) = \mathrm{N}(x, 0.25)(y),$$

and for $k = 2$, we used the proposal distribution

$$q_2(x, y) = \mathrm{N}_2(x, \Sigma)(y), \qquad \text{with} \quad \Sigma = \begin{pmatrix} 1.0 & 0 \\ 0 & 1.0 \end{pmatrix}.$$

We then applied a hill-climbing process to the end values by using the quasi-Newton algorithm to find the local minimum over x of $-\ln(\pi(k, x))$. The hill-climbing process took 24 iterations on average to converge. We repeated the procedure of simulated annealing and hill-climbing 100 times. For each run, we used random starting values $X = (X^{(1)})$ where $X^{(1)} \sim \mathrm{U}(-10, 10)$ for $k = 1$ and $X = (X^{(1)}, X^{(2)})$ where $X^{(1)}$ and $X^{(2)}$ are independent $\mathrm{U}(-10, 10)$ variates for $k = 2$. These runs produced 100 mode locations for each k.

2. Clustering

   In each dimension k, we define the distance between two mode locations to be their scaled Euclidean distance, using different scaling values s_k for each dimension, as defined in section 4.1.1. Then we clustered the mode locations from the initial exploration stage using the hierarchical clustering algorithm described in section 4.1.1, with $\xi = 0.01$. Consequently, the mode locations were reduced to

$m_1 = 2$ and $m_2 = 3$, and therefore $m = 5$ in total. The values of the corresponding mode locations $(1, \eta_1), \ldots, (2, \eta_5)$ turned out to be very close to the true mode locations (1,(5)), (1,(−3)), (2,(0, 0)), (2,(7, 0)) and (2,(0, −8)) respectively, with errors smaller than 1.0×10^{-8}.

3. Modelling

   At each of the mode locations (k_i, η_i), $i = 1, \ldots, m$, we fitted a k_i-variate normal distribution with center (k_i, η_i) and covariance matrix Σ_i of size $k_i \times k_i$. Each Σ_i is approximated numerically using the method described for the fixed dimension in section 4.1.2. Here we chose $v = 1 \times 10^{-5}$. The approximate normal distribution fitted at each (k_i, η_i) is very close to the true distribution at that mode.

4. Sampling

   In the sampling stage, we alternate between two parts: 1) local changes via 5 steps of the usual Metropolis-Hastings sampling method; and 2) a mode jumping step. The total of these 6 steps we call one iteration. In the first part, local changes are proposed via the usual Metropolis-Hastings algorithm by adding a small perturbation to the current value, i.e., by proposing a value $y = x + \epsilon$. For $k = 1$, we choose $\epsilon \sim \mathrm{N}(0, \varsigma_1^2)$ and for $k = 2$, we choose $\epsilon \sim \mathrm{N}_2(\mathbf{0}, \Sigma)$ where

$$\Sigma = \begin{pmatrix} \varsigma_2^2 & 0 \\ 0 & \varsigma_3^2 \end{pmatrix}.$$

   We can choose ς_i^2, $i = 1, \ldots, 3$ by referring to the local models fitted in the modelling stage. For ς_1^2 we choose the fitted model in $k = 1$ with the smallest variance and divide the standard deviation by two, to obtain $\varsigma_1^2 = 0.25^2$. For Σ, we choose $\varsigma_2^2 = 0.16^2$ and $\varsigma_3^2 = 0.16^2$, taking the smallest diagonal elements of the covariance matrices for the fitted models in $k = 2$ and dividing these by 4.

   In the second part, we define our "nearest mode" function $h(k, x) \in \{1, \ldots, m\}$ so that it chooses the nearest of the mode locations (k_i, η_i), with $k_i = k$, to the state (k, x) using the scaled Euclidean distance. Note that for each k, we use the corresponding scaling values s_k as defined in section 4.1.1 but only using the m_k mode locations found from the clustering stage. The mode jumping step then proceeds as given in section 7.2.4 for the MJM1 method, where we choose to jump to a different mode with equal probability, i.e., $p_{ij} = 1/4$ for $j = 1, \ldots, 5$, $j \neq i$.

7.4.2 Results for Example 7

When looking at the results of our mode jumping sampler we shall use the criteria as described in section 4.2.2. In doing this, we define the nearest mode function $m(k, x)$ so

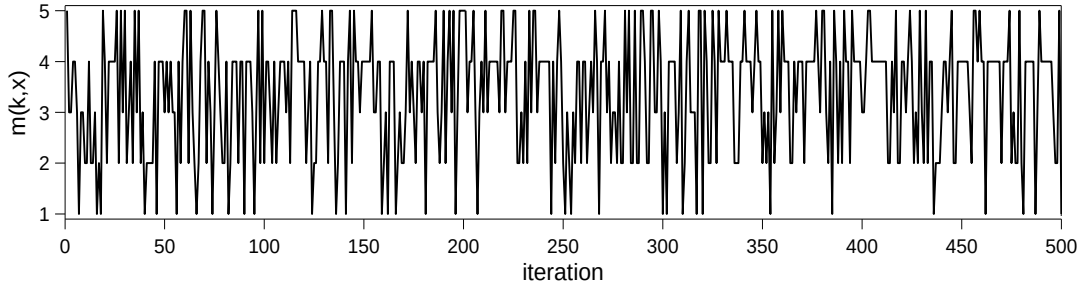


Figure 7-2: Plot of the values of $m(k, x)$ for the first 500 iterations, where the chain is simulated using the MJM1 method.

that it will choose the nearest mode j to (k, x) among the real mode locations (k_i, μ_i), for all k_i where $k_i = k$, using unscaled Euclidean distance.

In the sampling stage, we simulated a chain of length 1,000,000 using the MJM1 method. The values of $m(k, x)$ for the first 500 iterations are shown in Figure 7-2. From this figure, we can see that the chain simulated using our method is mixing well. This is also reflected from the estimate from these 1,000,000 iterations of the transition matrix for the movement of the chain between the five modes which is given by

$$\widehat{P} = \begin{pmatrix} 0.000 & 0.249 & 0.249 & 0.251 & 0.251 \\ 0.107 & 0.227 & 0.167 & 0.248 & 0.251 \\ 0.159 & 0.251 & 0.090 & 0.250 & 0.250 \\ 0.065 & 0.150 & 0.100 & 0.536 & 0.149 \\ 0.108 & 0.250 & 0.167 & 0.250 & 0.225 \end{pmatrix}.$$

The large values off the diagonal in $\widehat{P}$ indicate a high rate of jumping between modes.

The overall results for the MJM1 method are summarized in Table 7.1. From Table 7.1, we can see that our method successfully jumps to a different mode 70% of the time and successfully jumps between the two dimensions 41% of the time. The MJM1 method also produces a small value of λ^*, therefore convergence to the correct distribution over the five modes is fast for this method. We can see that the values of the IAC, $\tau(f)$, for $f^{(i)}(k, x) = I\{m(k, x) = i\}$, $i = 1, \ldots, 5$, which are estimated using $\widehat{P}$, are small and close to 1. This means that the MJM1 method has good estimation performance.

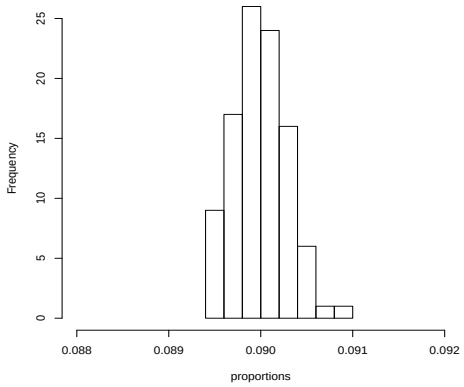
We can estimate the variance of the probability associated with each mode by simulating 100 repeated runs of the chain, each of length 1,000,000, and finding the proportion of time each chain spends in each mode. A histogram of the proportions for each mode is shown in Figure 7-3. From this figure, we can see that the estimated

Mode jumping rate (%)	70.51
Dimension jumping rate (%)	41.54
Total function calls ($\times 10^6$)	
a) initial exploration	0.025
b) sampling	
i) local steps	5
ii) mode jumping step	1
Eigenvalue λ_k of $\widehat{P}$ for which $\lvert\lambda_k\rvert = \lambda^*$	0.286
Estimated IAC for $f^{(i)} = I\{m(k,x) = i\}$ calculated using $\widehat{P}$	
1) $\hat{\tau}(f^{(1)})$	0.84
2) $\hat{\tau}(f^{(2)})$	1.08
3) $\hat{\tau}(f^{(3)})$	0.92
4) $\hat{\tau}(f^{(4)})$	1.80
5) $\hat{\tau}(f^{(5)})$	1.07

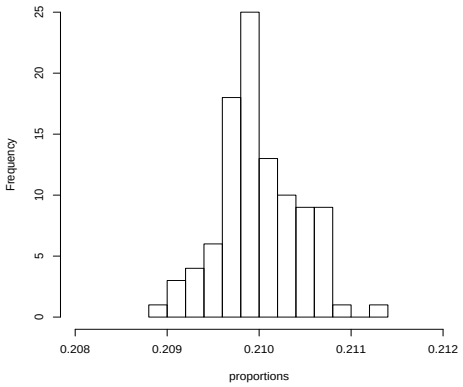
Table 7.1: Overall results for a chain of run length 1,000,000, simulated using the MJM1 method.

probability associated with each mode is very close to the true value, which is 0.09 for mode 1, 0.21 for mode 2, 0.14 for mode 3, 0.35 for mode 4 and 0.21 for mode 5. This indicates that the variance of these estimates are small. If we use these estimates to estimate $\tau(f^{(i)})$ we obtain $\hat{\tau}(f^{(1)}), \ldots, \hat{\tau}(f^{(5)})$ equal to 1.07, 1.29, 0.99, 1.79 and 1.08 respectively with standard errors of 0.1 (following the calculation in section 4.2.3). These are consistent with the values calculated from $\widehat{P}$.

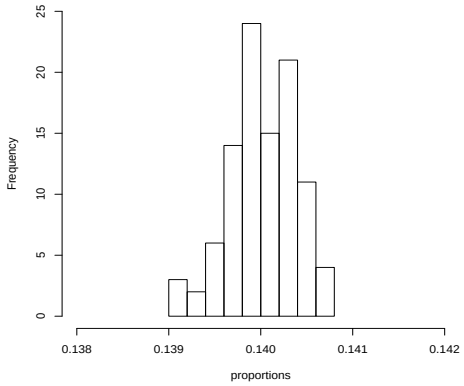
In general, the MJM1 method performs well when applied to this particular example. Our method does have an extra setup cost which needs to be considered in any efficiency comparisons. In the initial exploration stage, for each dimension k our method uses one call to the function π in each of the 100 iterations of the simulated annealing and 24 calls to the function π in the hill-climbing process, while in the sampling stage this method uses one call to the function π in each iteration. Since the total number of function calls used in the initial exploration stage is only 0.4% of the total number used in the sampling stage, this initial exploration is relatively cheap and accounting for it makes little difference to any efficiency comparisons. It is important that we run the initial exploration stage long enough to ensure that the probability of missing a mode is low. It is clear that we could increase the number of multiple short runs in the initial exploration stage substantially to reduce the probability of missing a mode, with little effect on the total computation.



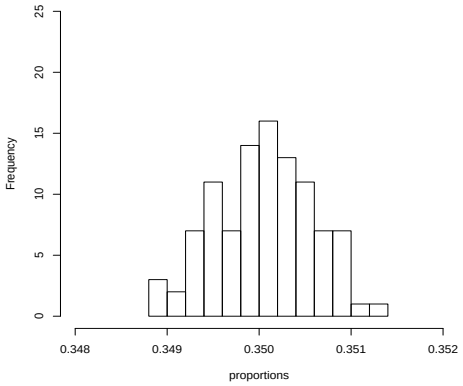
(a) Mode location = $(1, (5))$



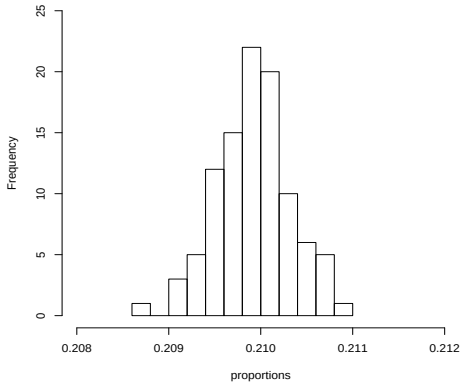
(b) Mode location = $(1, (\text{-}3))$



(c) Mode location = $(2, (0, 0))$



(d) Mode location = $(2, (7, 0))$



(e) Mode location = $(2, (0, -8))$

Figure 7-3: Histograms for the proportion of time the chains spend in each mode, where the chains are simulated using the MJM1 method.

7.5 Application to the problem of autoregressive model choice

7.5.1 Autoregressive model choice

We are interested in applying our mode jumping approach to the problem of model choice for autoregressive time series models of unknown order. This problem was used by Brooks et al. (2003) to illustrate their method of creating efficient proposal distributions when using RJMCMC. Getting an efficient MCMC sampler that mixes well between dimensions is usually difficult, as there is commonly no natural way to choose good jump proposals. The method of Brooks et al. (2003) tries to overcome this problem by fine-tuning the proposal distribution, where in every dimension jumping proposal it attempts to choose a sensible location and scale for the proposal distribution conditional on the current state.

We shall adapt the notation used by Brooks et al. (2003). Suppose we have data $x_1, \ldots, x_T$ from an autoregressive process of unknown order. We shall consider autoregressive models of order 1 to $k_{\max}$. For simplicity we consider observations x_1 to $x_{k_{\max}}$ as fixed for all models and specify the k^{th}-order autoregressive process by

$$X_t = \sum_{i=1}^{k} a_i X_{t-i} + \varepsilon_t, \qquad t = k_{\max}+1, \ldots, T,$$

where the $\varepsilon_t \sim N(0, \sigma_\varepsilon^2)$ and are i.i.d. We take the Bayesian approach, treating both the order K of the autoregressive process and the parameters $(a_1, \ldots, a_K)$ and σ_ε^2 as random variables. The distribution, π, from which we wish to sample is the posterior distribution of K, $(a_1, \ldots, a_K)$ and σ_ε^2. We denote the prior distribution of K by $p(K)$. Let $\boldsymbol{a}_K = (a_1, \ldots, a_K)^{\mathrm{T}}$ and $\boldsymbol{\theta}_K = (a_1, \ldots, a_K, \sigma_\varepsilon^2)$, then the prior density for $\boldsymbol{\theta}_K$ given $K = k$ is denoted by $p_k(\boldsymbol{\theta}_k)$. We denote the data likelihood under the model with $K = k$ and parameters $\boldsymbol{\theta}_k$ by $L_k(\boldsymbol{x}|\boldsymbol{\theta}_k)$. It follows that the posterior distribution π has sub-density for $K = k$

$$\pi(k, \boldsymbol{\theta}_k) = c\, L_k(\boldsymbol{x}|\boldsymbol{\theta}_k)\, p_k(\boldsymbol{\theta}_k)\, p(K = k), \qquad k = 1, \ldots, k_{\max},$$

where the value of c is that which makes this a probability distribution.

Following Brooks et al. (2003), we assume a uniform prior for K over the set $\{1, \ldots, k_{\max}\}$, independent $N(0, \sigma_a^2)$ priors for the coefficients a_i, $i = 1, \ldots, K$, and an inverse-Gamma(γ_1,γ_2) prior for σ_ε^2. The data likelihood given $K = k$ and $\boldsymbol{\theta}_k$ is

$$L_k(\boldsymbol{x}|\boldsymbol{\theta}_k) \;=\; \prod_{t=k_{\max}+1}^{T} \frac{1}{\sqrt{2\pi\sigma_\varepsilon^2}} \exp\left\{ -\frac{1}{2} \left(\frac{x_t - \sum_{i=1}^{k} a_i x_{t-i}}{\sigma_\varepsilon^2} \right) \right\}.$$

Let $r = T - k_{\max}$, define the $r \times 1$ vector $\boldsymbol{y} = (x_{k_{\max}+1}, \ldots, x_T)^{\mathrm{T}}$ and let Y_k be the $r \times k$ matrix

$$Y_k = \begin{pmatrix} x_{k_{\max}+1-1} & \cdots & x_{k_{\max}+1-k} \\ \vdots & \ddots & \vdots \\ x_{T-1} & \cdots & x_{T-k} \end{pmatrix},$$

for each $k = 1, \ldots, k_{\max}$. With these choices, the sub-density of π associated with $K = k$ is

$$\begin{aligned} \pi(k, \boldsymbol{\theta}_k) &= c\, L_k(\boldsymbol{x}|\boldsymbol{\theta}_k)\, p_k(\boldsymbol{\theta}_k)\, p(K = k) \\ &= c\, (2\pi\sigma_\varepsilon^2)^{-r/2} \exp\left\{-\frac{1}{2\sigma_\varepsilon^2}(\boldsymbol{y} - Y_k\boldsymbol{a}_k)^{\mathrm{T}}(\boldsymbol{y} - Y_k\boldsymbol{a}_k)\right\} \cdot \\ &\quad (2\pi\sigma_a^2)^{-k/2} \exp\left\{-\frac{1}{2\sigma_a^2}\boldsymbol{a}_k^{\mathrm{T}}\boldsymbol{a}_k\right\} \frac{\gamma_1^{\gamma_2}}{\Gamma(\gamma_1)}(\sigma_\varepsilon^2)^{-(\gamma_1+1)} \exp\left\{-\frac{\gamma_2}{\sigma_\varepsilon^2}\right\} \frac{1}{k_{\max}} \\ &= c\, \frac{\gamma_1^{\gamma_2}}{k_{\max} \cdot \Gamma(\gamma_1)}\, (2\pi)^{-(r+k)/2}\, \sigma_a^{-k}\, (\sigma_\varepsilon^2)^{-(r/2+\gamma_1+1)} \exp\left\{-\frac{1}{2\sigma_a^2}\boldsymbol{a}_k^{\mathrm{T}}\boldsymbol{a}_k - \right. \\ &\quad \left. \frac{1}{\sigma_\varepsilon^2}\left[\gamma_2 + \frac{1}{2}(\boldsymbol{y} - Y_k\boldsymbol{a}_k)^{\mathrm{T}}(\boldsymbol{y} - Y_k\boldsymbol{a}_k)\right]\right\}, \end{aligned} \tag{7.5.1}$$

for each $k = 1, \ldots, k_{\max}$.

We can find the conditional distributions of the parameters $\boldsymbol{a}_k$ and σ_ε^2 by using equation (7.5.1). For each $k = 1, \ldots, k_{\max}$, let I_k be the $k \times k$ identity matrix, define $Q_k = (\sigma_\varepsilon^2/\sigma_a^2)\, I_k$, $W_k = (Q_k + Y_k^{\mathrm{T}}Y_k)^{-1}$ and $\hat{\boldsymbol{a}}_k = W_k Y_k^{\mathrm{T}}\boldsymbol{y}$. It can be shown that, given $K = k$, the conditional distributions of σ_ε^2 given $\boldsymbol{a}_k$ and $\boldsymbol{a}_k$ given σ_ε^2 are

$$\sigma_\varepsilon^{-2}|\boldsymbol{a}_k \sim \mathrm{Gamma}\,(\frac{r}{2} + \gamma_1,\, \frac{1}{2}(\boldsymbol{y} - Y_k\boldsymbol{a}_k)^{\mathrm{T}}(\boldsymbol{y} - Y_k\boldsymbol{a}_k) + \gamma_2)$$

and

$$\boldsymbol{a}_k|\sigma_\varepsilon^2 \sim \mathrm{N}_k(\hat{\boldsymbol{a}}_k,\, \sigma_\varepsilon^2 W_k).$$

7.5.2 Example 8: Southern Oscillation Index data

Brooks et al. (2003) report on fitting the autoregressive model of section 7.5.1 to the Southern Oscillation Index (SOI) data set. These data consist of $T = 540$ monthly observations of the southern oscillation index during 1950–1995, where x is the "difference of the departure from the long-term monthly mean sea-level pressures at Tahiti in the South Pacific and Darwin in Northern Australia". The values are shown in Figure 7-4 and tabulated in Appendix E.

We follow Brooks et al. (2003) in assuming that this data set was generated by an autoregressive process of unknown order K in the range 1 to $k_{\max} = 10$, taking the priors for the Bayesian model defined in section 7.5.1 with $\sigma_a^2 = 1$, $\gamma_1 = 10^{-3}$ and

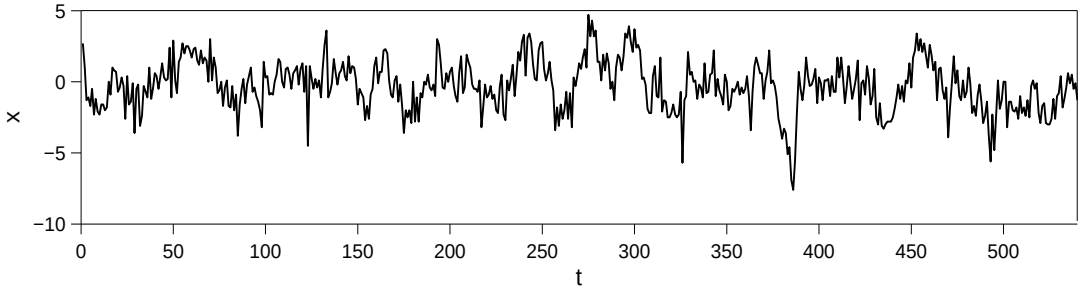


Figure 7-4: Plot of the Southern Oscillation Index (SOI) data, x, against time, t.

$\gamma_2 = 10^{-3}$. The posterior distribution π is then a sum of sub-densities on $\mathbb{R}^{k+1}$, for each $k = 1, \ldots, 10$, as given by equation (7.5.1).

7.5.3 Application of the MJM2 method to Example 8

We apply our mode jumping approach to sample from the target distribution π defined by equation (7.5.1) using the MJM2 method. This method does not require explicit modelling of each mode found in the initial exploration and clustering stages, but uses a cycle of Gibbs sampling to obtain proposals when jumping to new modes, which is feasible using the conditional distributions previously derived.

Implementation

1. Initial exploration

   For each $k = 1, \ldots, 10$, we carried out a run of length 100 using the Gibbs sampler with simulated annealing; in each run we simulated, on the j^{th} cycle, from the conditional distribution of each parameter raised to the power $1/T(j)$:

$$\sigma^{-2}_{\varepsilon,T(j)} | \boldsymbol{a}_k \sim \text{Gamma}\,(\tfrac{\frac{r}{2}+\gamma_1+1}{T(j)} - 1,\ \tfrac{1}{2\,T(j)}(\boldsymbol{y} - Y_k\boldsymbol{a}_k)^{\mathrm{T}}(\boldsymbol{y} - Y_k\boldsymbol{a}_k) + \tfrac{\gamma_2}{T(j)}),$$

$$\boldsymbol{a}_{k,T(j)} | \sigma^2_\varepsilon \sim \mathrm{N}_k(\hat{\boldsymbol{a}}_k,\ \sigma^2_\varepsilon\, T(j)\, W_k).$$

   Here, the logarithmic temperature function, $T(j) = 1/\ln(1+j)$, was used. After running the Gibbs sampler with simulated annealing, we applied the hill-climbing process to the end values where we updated σ^2_ε and $\boldsymbol{a}_k$ by changing their values to the modal value of each conditional distribution. We repeated the process of updating the values of $\boldsymbol{a}_k$ and σ^2_ε until the values converged. On average, this took around five iterations. For each $k = 1, \ldots, 10$, we repeated the procedure of simulated annealing and hill-climbing 100 times. For each run, we used random starting values where $\sigma^2_\varepsilon \sim \mathrm{U}(0, 5)$ and $a_i \sim \mathrm{U}(-2, 2)$, $i = 1, \ldots, k$. These runs produced 100 mode locations for each k.

2. Clustering

We clustered the modes for each k following the procedure described in section 7.4.1, where we only found $m_k = 1$ mode for each $k = 1, \ldots, 10$; therefore, the total number of modes m is equal to 10. The values of the corresponding mode locations are given in Table 7.2.

3. Sampling

   In the sampling stage, we alternate between two parts: 1) local changes via 5 cycles of the Gibbs sampler; and 2) a mode jumping step. The total of these 6 steps we call one iteration. In the second part, we define our "nearest mode" function $h(k, \boldsymbol{\theta}_k) \in \{1, \ldots, m\}$ so that it chooses the nearest mode location (k_i, η_i), where $k_i = k$, to the state $(k, \boldsymbol{\theta}_k)$. Since we only found one mode for each $k = 1, \ldots, 10$, we have simply $h(k, \boldsymbol{\theta}_k) = k$. The sampling stage then proceeds as given in section 7.2.4 for the MJM2 method. Here we shall choose to jump to a different mode with equal probability, i.e., $p_{ij} = 1/9$ for $j \in \{1, \ldots, 10\}$, $j \neq i$.

Results for Example 8

When looking at the results of our approach in Example 8 we shall use the criteria as described in section 4.2.2. In doing this, we define the nearest mode function $m(k, \boldsymbol{\theta}_k) = k$. We simulated a chain of length 10 million using the MJM2 method. The values of $m(k, \boldsymbol{\theta}_k)$ for the first 1000 iterations are shown in Figure 7-5. From this figure, we can see that the chain simulated using the MJM2 method is mixing fairly well. The estimate from these 10 million iterations of the transition matrix for the movement of the chain between the ten modes, denoted by $\widehat{P}$, is equal to

$$\begin{pmatrix}
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.8314 & 0.1110 & 0.0547 & 0.0026 & 0.0001 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0448 & 0.9318 & 0.0222 & 0.0011 & 0.0001 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.1112 & 0.1112 & 0.7720 & 0.0052 & 0.0002 & 0.0001 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.1118 & 0.1130 & 0.1115 & 0.6568 & 0.0048 & 0.0015 & 0.0004 & 0.0001 & 0.0001 \\
0.0000 & 0.1080 & 0.1103 & 0.1175 & 0.0952 & 0.5172 & 0.0404 & 0.0102 & 0.0008 & 0.0004 \\
0.0000 & 0.1145 & 0.0974 & 0.1054 & 0.1386 & 0.1042 & 0.4089 & 0.0275 & 0.0011 & 0.0023 \\
0.0000 & 0.1158 & 0.1105 & 0.0947 & 0.0895 & 0.1263 & 0.1158 & 0.3316 & 0.0158 & 0.0000 \\
0.0588 & 0.1765 & 0.0000 & 0.2353 & 0.1765 & 0.0000 & 0.0588 & 0.0588 & 0.1765 & 0.0588 \\
0.0000 & 0.1000 & 0.2000 & 0.1000 & 0.1000 & 0.1000 & 0.1000 & 0.1000 & 0.2000 & 0.0000
\end{pmatrix}.$$

Note that in the 10 million iterations the chain actually visited $K = 1$ one time, $K = 9$ 17 times and $K = 10$ ten times. This means that a chain of length 10 million is reasonable, since there are states which we do not visit very often and we are trying to find out more about them. This also means that we should be cautious about the rows of $\widehat{P}$ associated with states that were visited only a few times.

The overall results for the MJM2 method are summarized in Table 7.3. From

i	k_i	$\boldsymbol{\eta}_i$	
		σ_ε^2	$a_1, \ldots, a_{k_i}$
1	1	1.808	0.648
2	2	1.685	0.478, 0.261
3	3	1.659	0.446, 0.202, 0.123
4	4	1.650	0.437, 0.186, 0.089, 0.076
5	5	1.649	0.436, 0.185, 0.086, 0.069, 0.016
6	6	1.649	0.435, 0.184, 0.085, 0.067, 0.010, 0.012
7	7	1.636	0.436, 0.185, 0.091, 0.074, 0.027, 0.050, −0.088
8	8	1.627	0.430, 0.189, 0.093, 0.080, 0.034, 0.065, −0.054, −0.077
9	9	1.625	0.427, 0.187, 0.095, 0.081, 0.036, 0.068, −0.048, −0.063, −0.032
10	10	1.606	0.423, 0.180, 0.090, 0.088, 0.041, 0.076, −0.038, −0.042, 0.014, −0.107

Table 7.2: Values of the mode locations $(k_i, \boldsymbol{\eta}_i)$ for the Southern Oscillation Index data.

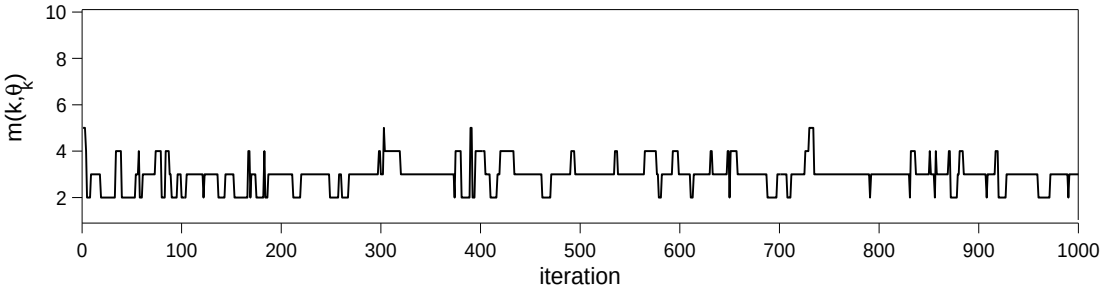


Figure 7-5: Plot of the values of $m(k, \boldsymbol{\theta}_k)$ for the first 1000 iterations, where the chain is simulated using the MJM2 method.

this table we can see that this method successfully jumps to a different mode and a different dimension nearly 11.5% of the time. The MJM2 method also produces λ^*=0.821, therefore convergence to the correct distribution over the ten modes is fairly fast. We can see that the values of the IAC, $\tau(f^{(i)})$, for $f^{(i)}(k, \boldsymbol{\theta}_k) = I\{m(k, \boldsymbol{\theta}_k) = i\}$, $i = 1, \ldots, 10$, which are estimated using $\widehat{P}$, are small. This means that the MJM2 method has good estimation performance. Note that the chain simulated by this method visited all $K = 1, \ldots, 10$, but spent a high proportion of the time in $K = 3$ (around 62%).

In general, the MJM2 method performs well when applied to this particular example. In this case, there are two types of computation used: 1) the computation used in one cycle of the Gibbs sampler; and 2) the computation used to evaluate $\pi(k', \boldsymbol{\theta}_{k'})/\pi(k, \boldsymbol{\theta}_k)$ in the acceptance probability $\alpha((k, \boldsymbol{\theta}_k), (k', \boldsymbol{\theta}_{k'}))$. In the initial exploration stage this method uses 1 cycle of the Gibbs sampler in each iteration of the simulated annealing, while in the sampling stage it uses 7 cycles of the Gibbs sampler and one evaluation of $\pi(k', \boldsymbol{\theta}_{k'})/\pi(k, \boldsymbol{\theta}_k)$ in each iteration. Since the amount of computation used in the initial exploration stage is a small fraction of the total amount used in the sampling stage, this initial exploration is relatively cheap and accounting for it makes little difference to any efficiency comparisons. On the other hand, we do have to run the initial exploration stage long enough to ensure that the probability of missing a mode is low. In our initial exploration, we do multiple short runs, using random starting values that should allow separate runs of the initial exploration to reach different modes.

Note that we applied the MJM2 method to this example as Gibbs sampling is feasible using the conditional distributions previously derived. However, in some problems it might not be possible to use the Gibbs sampler. In these problems the MJM1 method, wherein we produce a randomly sampled value at a mode by sampling from an approximate distribution already fitted at that mode, might be applicable. We can see how the MJM1 method perform on this example in the next section.

Mode jumping rate (%)	11.48
Dimension jumping rate (%)	11.48
Total computation ($\times 10^7$)	
1) initial exploration	
a) Gibbs cycles	0.01
2) sampling	
a) local steps	
i) Gibbs cycles	5
b) mode jumping step	
i) Gibbs cycles	2
ii) evaluations of $\pi(k', \boldsymbol{\theta}_{k'})/\pi(k, \boldsymbol{\theta}_k)$	1
Eigenvalue λ_k of $\widehat{P}$ for which $\lvert\lambda_k\rvert = \lambda^*$	0.821
Estimated IAC for $f^{(i)} = I\{m(k, \boldsymbol{\theta}_k) = i\}$ calculated using $\widehat{P}$	
1) $\hat{\tau}(f^{(1)})$	1.00
2) $\hat{\tau}(f^{(2)})$	8.33
3) $\hat{\tau}(f^{(3)})$	10.14
4) $\hat{\tau} f^{(4)})$	6.99
5) $\hat{\tau}(f^{(5)})$	4.85
6) $\hat{\tau}(f^{(6)})$	3.24
7) $\hat{\tau}(f^{(7)})$	2.48
8) $\hat{\tau}(f^{(8)})$	2.04
9) $\hat{\tau}(f^{(9)})$	1.47
10) $\hat{\tau}(f^{(10)})$	1.03
Estimated probability associated with mode:	
1	0.0000001
2	0.2504578
3	0.6197899
4	0.1235807
5	0.0057978
6	0.0002647
7	0.0000873
8	0.0000190
9	0.0000017
10	0.0000010

Table 7.3: Overall results for a chain of run length 10,000,000, simulated using the MJM2 method.

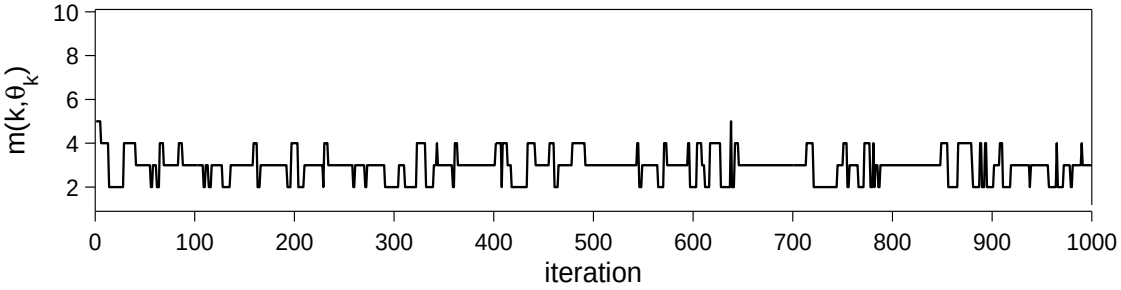


Figure 7-6: Plot of the values of $m(k, \boldsymbol{\theta}_k)$ for the first 1000 iterations, where the chain is simulated using the MJM1 method.

7.5.4 Application of the MJM1 method to Example 8

Implementation

Implementation of the initial exploration and clustering stages remain the same as described in 7.5.3. In the modelling stage, at each of the mode locations $(k_i, \boldsymbol{\eta}_i)$, $i = 1, \ldots, 10$, we fitted a $(k_i + 1)$-variate normal distribution with center $(k_i, \boldsymbol{\eta}_i)$ and covariance matrix Σ_i of size $(k_i + 1) \times (k_i + 1)$. Each Σ_i is approximated numerically using the method described for the fixed dimension in section 4.1.2, where we chose $v = 0.01$. With a multivariate normal model for $\boldsymbol{\theta}_k$ at each mode, sampling is conducted by running a Markov chain in the usual way for the MJM1 method.

Results for Example 8

We simulated a chain of length 10 million using the MJM1 method. The values of $m(k, \boldsymbol{\theta}_k)$ for the first 1000 iterations are shown in Figure 7-6. From this figure, we can see that the chain simulated using this method is also mixing fairly well. The estimate from these 10 million iterations of the transition matrix for the movement of the chain between the ten modes, denoted by $\widehat{P}'$, is equal to

$$\begin{pmatrix}
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.8325 & 0.1105 & 0.0543 & 0.0026 & 0.0001 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0444 & 0.9324 & 0.0222 & 0.0010 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.1093 & 0.1113 & 0.7739 & 0.0051 & 0.0002 & 0.0001 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.1118 & 0.1115 & 0.1117 & 0.6579 & 0.0050 & 0.0018 & 0.0002 & 0.0000 & 0.0000 \\
0.0000 & 0.1117 & 0.1141 & 0.1090 & 0.1082 & 0.5081 & 0.0422 & 0.0055 & 0.0012 & 0.0000 \\
0.0000 & 0.0953 & 0.1183 & 0.1204 & 0.1152 & 0.1016 & 0.4209 & 0.0272 & 0.0010 & 0.0000 \\
0.0000 & 0.1023 & 0.1080 & 0.0852 & 0.0966 & 0.0966 & 0.1250 & 0.3807 & 0.0000 & 0.0057 \\
0.0000 & 0.0000 & 0.0833 & 0.1667 & 0.2500 & 0.2500 & 0.0833 & 0.0000 & 0.0000 & 0.1667 \\
0.0000 & 0.1667 & 0.1667 & 0.0000 & 0.5000 & 0.1667 & 0.0000 & 0.0000 & 0.0000 & 0.0000
\end{pmatrix}.$$

We can see that the values of $\widehat{P}'$ are very similar to those of $\widehat{P}$ for the MJM2 method, given in section 7.5.3.

The overall results for the MJM1 method are summarized in Table 7.4. These are very similar to the results for the MJM2 method as given in Table 7.3. However, each mode jumping step in the MJM2 method involves 2 cycles of the Gibbs sampler and an evaluation of $\pi(k', \boldsymbol{\theta}_{k'})/\pi(k, \boldsymbol{\theta}_k)$. Each mode jumping step in the MJM1 method, on the other hand, involves generating a random sample from a multivariate normal distribution and evaluating

$$\frac{g_{h(k,\theta_k)}(k, \boldsymbol{\theta}_k)}{g_{h(k',\theta_{k'})}(k', \boldsymbol{\theta}_{k'})} = \frac{\mathrm{N}_{k+1}((k, \boldsymbol{\eta}_{h(k,\theta_k)}), \Sigma_{h(k,\theta_k)})(\boldsymbol{\theta}_k)}{\mathrm{N}_{k'+1}((k', \boldsymbol{\eta}_{h(k',\theta_{k'})}), \Sigma_{h(k',\theta_{k'})})(\boldsymbol{\theta}_{k'})},$$

apart from evaluating $\pi(k', \boldsymbol{\theta}_{k'})/\pi(k, \boldsymbol{\theta}_k)$. In this case, the MJM1 method has a computational advantage since the means and covariance matrices of the fitted normal distributions are already known before sampling, whereas in the MJM2 method the mean and covariance matrix of the conditional distribution of $\boldsymbol{a}_k$ given σ_ε^2, which is k-variate normal, have to be calculated in each cycle of the Gibbs sampler. Furthermore, in our experience, the MJM1 method runs considerably faster than the MJM2 method — around 5 times as fast.

Note that for the application of both the MJM2 and MJM1 methods we use the Gibbs sampler when proposing local movements. We chose this way of moving locally because the conditional distributions of the parameters are tractable and easy to sample from. Another option, particularly when Gibbs sampling is not feasible, is to propose local changes via the Metropolis-Hastings algorithm by adding a small perturbation to the current value. An example of how we can use the models fitted during the modelling stage to specify the distributions for these small perturbations was described in section 7.4.1.

7.5.5 Application of the MJM1 method to Example 8 using estimated weights

Implementation

When we applied the MJM method to Examples 7 and 8 we chose to jump to a different mode with equal probability, i.e., $p_{ij} = 1/(m-1)$ for $j \in \{1, \ldots, m\}$, $j \neq i$. However, we can also specify p_{ij} using the weights w_j as described in section 7.2.4. For the explicit modelling case these weights can be estimated using the approximate local distribution at the mode locations as described in section 4.1.2. Alteratively, we could estimate the w_j by using samples from a short run of our mode jumping method.

We now apply the MJM1 method to Example 8 using the weights w_j estimated using the second approach. We ran the MJM1 method as described in section 7.5.4

Mode jumping rate (%)	11.38
Dimension jumping rate (%)	11.38
Total computation ($\times 10^7$)	
1) initial exploration	
a) Gibbs cycles	0.01
2) sampling	
a) local steps	
i) Gibbs cycles	5
b) mode jumping step	
i) evaluations of $g_{h(k,\theta_k)}(k, \boldsymbol{\theta}_k)/g_{h(k',\theta_{k'})}(k', \boldsymbol{\theta}_{k'})$	1
ii) evaluations of $\pi(k', \boldsymbol{\theta}_{k'})/\pi(k, \boldsymbol{\theta}_k)$	1
Eigenvalue λ_k of $\widehat{P}'$ for which $\lvert\lambda_k\rvert = \lambda^*$	0.822
Estimated IAC for $f^{(i)} = I\{m(k, \boldsymbol{\theta}_k) = i\}$ calculated using $\widehat{P}'$	
1) $\hat{\tau}(f^{(1)})$	–
2) $\hat{\tau}(f^{(2)})$	8.40
3) $\hat{\tau}(f^{(3)})$	10.20
4) $\hat{\tau}(f^{(4)})$	7.05
5) $\hat{\tau}(f^{(5)}$	4.87
6) $\hat{\tau}(f^{(6)})$	3.16
7) $\hat{\tau}(f^{(7)})$	2.55
8) $\hat{\tau}(f^{(8)})$	2.27
9) $\hat{\tau}(f^{(9)})$	1.00
10) $\hat{\tau}(f^{(10)})$	1.00
Estimated probability associated with mode:	
1	0.0000000
2	0.2492139
3	0.6210700
4	0.1236901
5	0.0056578
6	0.0002533
7	0.0000955
8	0.0000176
9	0.0000012
10	0.0000006

Table 7.4: Overall results for a chain of run length 10,000,000, simulated using the MJM1 method.

for 10,000 iterations to obtain the proportion of time the chain spent in each mode; we obtained 0.23, 0.63, 0.13 and 0.01 for modes 2, 3, 4 and 5 respectively, and zero for the rest. However, we still want to have at least some small probability of visiting every mode. Therefore we choose w_2, w_3 and w_4 equal 0.23, 0.57 and 0.13 respectively, while setting the rest of the w_j equal to 0.01. Implementation of the initial exploration, clustering, modelling and sampling stages remains as described in 7.5.4, except that in the sampling stage we specify p_{ij} using the weights w_j as described in section 7.2.4, i.e., for each i,

$$p_{i,j} = \frac{w_j}{\sum_{l \neq i} w_l} \qquad \text{for } j = 1, \ldots, 10 \text{ and } j \neq i.$$

Results for Example 8

We simulated a chain of length 10 million using the weighted MJM1 method. The values of $m(k, \boldsymbol{\theta}_k)$ for the first 1000 iterations are shown in Figure 7-7. From this figure, we can see that the chain simulated using this method is mixing much better than the chain from the un-weighted method as shown in Figure 7-6. The estimate from these 10 million iterations of the transition matrix for the movement of the chain between the ten modes, denoted by $\widehat{P}''$, is equal to

$$\begin{pmatrix}
0.0000 & 0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.1416 & 0.7288 & 0.1241 & 0.0053 & 0.0002 & 0.0001 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.2939 & 0.5709 & 0.1295 & 0.0054 & 0.0002 & 0.0001 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.2512 & 0.6474 & 0.0951 & 0.0059 & 0.0003 & 0.0001 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.2315 & 0.5735 & 0.1261 & 0.0683 & 0.0005 & 0.0002 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.2269 & 0.5793 & 0.1339 & 0.0117 & 0.0421 & 0.0049 & 0.0012 & 0.0000 & 0.0000 \\
0.0000 & 0.2126 & 0.5911 & 0.1396 & 0.0076 & 0.0109 & 0.0327 & 0.0055 & 0.0000 & 0.0000 \\
0.0000 & 0.2410 & 0.5692 & 0.1179 & 0.0103 & 0.0256 & 0.0103 & 0.0256 & 0.0000 & 0.0000 \\
0.0000 & 0.3750 & 0.4375 & 0.1875 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.5000 & 0.2000 & 0.3000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000
\end{pmatrix}.$$

Note that some of the zeroes in $\widehat{P}''$ are only zero on rounding. We can see that there are larger values off the diagonal in $\widehat{P}''$ when compared to the estimate for the un-weighted MJM1 method given by $\widehat{P}'$ in section 7.5.4.

The overall results for the weighted MJM1 method are summarized in Table 7.5. When we compare these with the results for the un-weighted MJM1 method given in Table 7.4, we can see that the weighted MJM1 method performs better than the un-weighted method as its mode jumping rate is 5 times higher and its values of λ^* and $\hat{\tau}(f^{(i)})$ are much smaller. In particular, the value of $\hat{\tau}(f^{(3)})$ for the weighted MJM1 method is one thirteenth of the value for the un-weighted method, i.e., this method obtains the same estimation accuracy as the previous method using just one thirteenth of the run length. If we take into account that the weighted MJM1 method costs

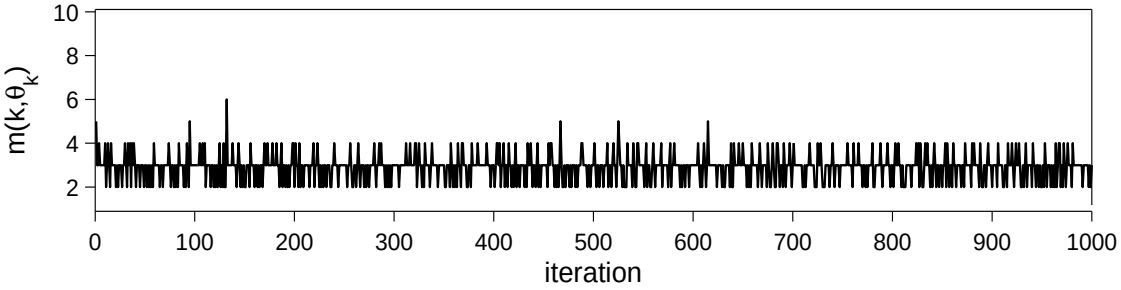


Figure 7-7: Plot of the values of $m(k, \boldsymbol{\theta}_k)$ for the first 1000 iterations, where the chain is simulated using the weighted MJM1 method.

just 0.1% more than the un-weighted MJM1 method, then overall the weighted MJM1 method is still 12.9 times more efficient than the un-weighted method, which is a high gain in efficiency.

We can compare the results of the weighted MJM1 method for Example 8 to the results obtained by the methods proposed by Brooks et al. (2003). The methods proposed by Brooks et al. (2003) managed to visit at least $K = 2, \ldots, 6$, and their methods attributed the highest posterior probability to $K = 3$ followed by $K = 2$, with steadily decreasing probabilities for larger values of K. These are similar to the results obtained by our method, as shown in Table 7.5. The posterior probability associated with $K = 3$ is estimated by Brooks et al. (2003) to be around 0.61, which is close to our estimate of 0.62.

The acceptance rate of dimension jumping moves for the MJM1 method, which is the mode jumping rate in Table 7.5, is much higher than the acceptance rate obtained by Brooks et al. (2003) for their methods. In fact, our mode jumping proposals will successfully jump between dimension at least three times as frequently. Furthermore, the proposals in the Brooks et al. (2003) method are set up so that jumps take place only between dimensions differing by one, while our proposals can jump between any pair of dimensions. Therefore the chain simulated by our method mixes better and faster between dimensions compared to their methods.

In term of cost, it is difficult to compare the costs of the MJM1 method and Brooks et al. (2003) methods directly. However, we can try to compare the computation used by each dimension jumping move. In their methods, Brooks et al. (2003) try to fine-tune the proposal distribution in every dimension jumping proposal, deriving a sensible location and scale for the proposal distribution analytically using the conditional densities of the parameters. Their methods then uses at least two evaluations of π, where these two evaluations are used to compute the acceptance probability of their dimension jumping move. The MJM1 method, on the other hand, uses *only* two

Mode jumping rate (%)	59.86
Dimension jumping rate (%)	59.86
Total computation ($\times 10^7$)	
1) initial exploration	
a) Gibbs cycles	0.01
2) initial short run	
a) local steps	
i) Gibbs cycles	0.005
b) mode jumping step	
i) evaluations of $g_{h(k,\theta_k)}(k,\boldsymbol{\theta}_k)/g_{h(k',\theta_{k'})}(k',\boldsymbol{\theta}_{k'})$	0.001
ii) evaluations of $\pi(k',\boldsymbol{\theta}_{k'})/\pi(k,\boldsymbol{\theta}_k)$	0.001
3) sampling	
a) local steps	
i) Gibbs cycles	5
b) mode jumping step	
i) evaluations of $g_{h(k,\theta_k)}(k,\boldsymbol{\theta}_k)/g_{h(k',\theta_{k'})}(k',\boldsymbol{\theta}_{k'})$	1
ii) evaluations of $\pi(k',\boldsymbol{\theta}_{k'})/\pi(k,\boldsymbol{\theta}_k)$	1
Eigenvalue λ_k of $\widehat{P}''$ for which $\lvert\lambda_k\rvert = \lambda^*$	-0.154
Estimated IAC for $f^{(i)} = I\{m(k,\boldsymbol{\theta}_k) = i\}$ calculated using $\widehat{P}''$	
1) $\hat{\tau}(f^{(1)})$	1.00
2) $\hat{\tau}(f^{(2)})$	0.75
3) $\hat{\tau}(f^{(3)})$	0.78
4) $\hat{\tau}(f^{(4)})$	0.94
5) $\hat{\tau}(f^{(5)}$	1.13
6) $\hat{\tau}(f^{(6)})$	1.09
7) $\hat{\tau}(f^{(7)})$	1.07
8) $\hat{\tau}(f^{(8)})$	1.05
9) $\hat{\tau}(f^{(9)})$	1.00
10) $\hat{\tau}(f^{(10)})$	1.00
Estimated probability associated with mode:	
1	0.0000001
2	0.2501485
3	0.6198537
4	0.1238640
5	0.0057727
6	0.0002472
7	0.0000917
8	0.0000195
9	0.0000016
10	0.0000010

Table 7.5: Overall results for a chain of run length 10,000,000, simulated using the weighted MJM1 method.

evaluations of π, which are used to compute the acceptance probability of our mode jumping move. Therefore in this respect the MJM1 method costs no more than Brooks et al. (2003) methods. Our method does have an extra setup cost which needs to be considered in efficiency comparisons. However, since the total number of function calls used in the initial exploration stage and the initial short run is a small fraction of the total number used in the sampling stage, the set up cost is relatively small and accounting for it makes little difference to any efficiency comparisons.

7.6 Summary and discussion

We have shown how the mode jumping approach can be applied to sample from a target distribution with variable dimension using a mixture of normal distributions in 2 dimensions. We have also shown how we can apply this approach to the problem of model choice for autoregressive time series models of unknown order. In both examples, we have shown that our mode jumping approach works well. We have also shown that with a very small additional cost our mode jumping method can be made much more efficient by using estimated weights in specifying the probabilities of each mode in the mode jumping steps.

If we compare our mode jumping approach to the Brooks et al. (2003) methods, then our approach has several advantages. The Brooks et al. (2003) methods need to find a good location and scale for the proposal distribution in every dimension jumping attempt. In the time series example, they use analytical results to find these values. However, in other problems, the same exercise may have to be done numerically, which means that more computation may be needed for their method. Our methods, on the other hand, will stay the same, and will use similar types of computation for different problems. Furthermore, our methods can move between any pair of dimensions, and doing this with a good set of weights increases the efficiency considerably.

Chapter 8

Conclusions and Future Work

We have shown that our new mode jumping approach is general and versatile as we can apply the approach to a variety of target distributions. It also shows a big improvement on efficiency compared to known MCMC sampling methods. Furthermore, it can be made more efficient through adaption to the particular problem. For example, in the image analysis problem in chapter 5 the MJD method can be made more efficient if we exploit the fact that "modes" in this problem involve certain features that appear or disappear and this can be achieved by only changing certain pixels for a jump between modes. Our mode jumping approach is also easy to apply with standard techniques used to implement each of the stages.

The examples that we have used throughout this thesis were chosen to be challenging to the commonly known MCMC sampling methods. However, these methods do still work when applied to these examples, for example, in chapter 5 the Gibbs sampler is able to sample from the posterior distribution of the true image albeit slowly. In the future, we might consider even more challenging applications. For example, in the image analysis problem our new mode jumping approach can be applied to larger images, which would involve larger objects/features that appear or disappear. In this case, the Gibbs sampler may be unable to move between the modes and may fail to sample correctly from the posterior distribution. In the variable dimension setting, more complex problems such as modelling a univariate density by a mixture of Gaussian distributions, as discussed by Green & Richardson (1997), will pose an interesting challenge to our mode jumping approach.

In conclusion, we believe that we have a useful contribution to a methodology. We have also demonstrated this methodology on a variety of problems. We believe it exhibits a real usefulness in simulation where we need to sample from complex, high-dimensional distributions which are not analytically tractable, particularly when these are multi-modal.

Appendix A

Data set: Soil phosphate

This data set consists of a 16×16 grid of measurements of log phosphate level taken at 10 m intervals in a Laconia Survey in Greece.

Log phosphate level (by row)							
1.68761	1.08610	1.01467	-0.13203	0.22184	0.61332	1.15507	1.31777
1.31777	1.01467	1.58092	-0.06853	-1.33357	1.25427	2.36426	2.59158
1.15507	1.01467	0.05272	0.61332	0.42714	0.70030	0.27517	-0.33559
0.61332	0.94058	-0.00698	0.74240	0.90249	1.76405	2.23024	2.43700
0.61332	1.37931	1.25427	0.05272	0.27517	0.74240	-0.06853	0.56838
1.25427	1.76405	1.18869	-0.40833	1.18869	0.82405	1.63498	2.36426
1.01467	0.37776	0.42714	1.37931	0.16704	1.43902	1.01467	0.61332
-0.91096	1.76405	1.37931	1.95467	2.02172	1.95467	1.83768	2.02172
1.15507	1.08610	1.76405	1.52536	-1.11112	1.76405	1.18869	1.18869
1.18869	1.25427	1.50000	1.50000	3.22398	1.68761	1.50000	1.43902
1.18869	1.43902	1.63498	1.88530	1.08610	0.56838	-0.19761	1.25427
0.27517	2.02172	1.50000	1.50000	1.43902	1.68761	1.50000	1.15507
0.74240	1.58092	2.23024	1.76405	0.70030	1.31777	0.42714	0.05272
0.70030	1.52536	1.25427	2.50718	1.18869	1.58092	0.90249	4.36716
1.43902	1.76405	0.82405	2.59158	2.75039	1.31777	1.15507	0.70030
1.68761	1.43902	2.92569	1.37931	-0.06853	0.82405	1.01467	0.82405
1.52536	1.52536	1.52536	2.02172	1.76405	1.43902	1.08610	1.63498
1.58092	1.68761	1.18869	0.05272	0.70030	0.82405	0.82405	2.23024
1.08610	2.65663	2.02172	2.82531	1.83768	1.43902	1.52536	1.83768
1.25427	2.28878	1.25427	1.25427	0.61332	1.15507	0.42714	-0.40833
1.18869	1.83768	2.08659	2.36426	1.76405	1.95467	1.37931	1.52536
-0.40833	1.63498	1.76405	1.68761	0.05272	1.08610	-0.19761	1.52536
1.01467	1.37931	2.08659	1.50000	1.95467	2.50718	1.83768	1.68761
0.56838	0.42714	1.15507	0.42714	1.08610	1.01467	0.70030	0.74240
0.94058	1.68761	2.02172	2.36426	1.58092	2.36426	1.88530	1.83768
0.94058	-0.00698	1.63498	-0.72901	1.18869	1.08610	0.70030	0.16704
1.08610	1.52536	1.63498	1.76405	1.58092	1.88530	1.88530	1.58092
0.94058	2.75039	1.08610	0.27517	1.31777	0.27517	1.01467	1.52536
1.25427	0.70030	1.43902	1.76405	2.14942	2.02172	1.68761	1.68761
0.90249	0.42714	-0.40833	1.43902	1.43902	1.52536	1.63498	1.37931
1.31777	1.01467	1.15507	1.58092	1.25427	1.58092	1.83768	1.15507
0.16704	0.22184	1.08610	1.43902	1.50000	1.50000	2.36426	1.83768

Table A.1: Values of the soil phosphate data

Appendix B

Data set: Star cluster CYG OB1

In this data set, two variables are observed in 47 stars in the direction of Cygnus. The independent variable (X) is the logarithm of the effective temperature at the surface of the stars and the dependent variable (Y) is the logarithm of the light intensity. The values are provided by Rousseeuw & Leroy (1987).

Index of star, i	y_i	x_i	Index of star, i	y_i	x_i
1	5.23	4.37	25	5.02	4.38
2	5.74	4.56	26	4.66	4.42
3	4.93	4.26	27	4.66	4.29
4	5.74	4.56	28	4.90	4.38
5	5.19	4.30	29	4.39	4.22
6	5.46	4.46	30	6.05	3.48
7	4.65	3.84	31	4.42	4.38
8	5.27	4.57	32	5.10	4.56
9	5.57	4.26	33	5.22	4.45
10	5.12	4.37	34	6.29	3.49
11	5.73	3.49	35	4.34	4.23
12	5.45	4.43	36	5.62	4.62
13	5.42	4.48	37	5.10	4.53
14	4.05	4.01	38	5.22	4.45
15	4.26	4.29	39	5.18	4.53
16	4.58	4.42	40	5.57	4.43
17	3.94	4.23	41	4.62	4.38
18	4.18	4.42	42	5.06	4.45
19	4.18	4.23	43	5.34	4.50
20	5.89	3.49	44	5.34	4.45
21	4.38	4.29	45	5.54	4.55
22	4.22	4.29	46	4.98	4.45
23	4.42	4.42	47	4.50	4.42
24	4.85	4.49			

Table B.1: Values of the star cluster data

Appendix C

Derivation of the marginal posterior distribution of $\boldsymbol{\delta}$

In general:
If $\boldsymbol{Z} \sim \mathrm{N}_p(\boldsymbol{\mu}, \Sigma)$, $f(z) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{\mu})^{\mathrm{T}}\Sigma^{-1}(\boldsymbol{z}-\boldsymbol{\mu})\right\}$.
If $Z \sim \mathrm{Beta}\,(a,b)$, $f(z) = \frac{\Gamma(a+b)}{\Gamma(a)\,\Gamma(b)} z^{a-1}(1-z)^{b-1}$, where $0 \leq z \leq 1$, $a > 0$, $b > 0$.
If $Z \sim \mathrm{Inv-Gamma}\,(a,b)$, $f(z) = \frac{b^a}{\Gamma(a)} z^{-(a+1)} e^{-b/z}$, where $z, a, b > 0$.

The posterior distribution of $(\boldsymbol{\beta}, \sigma^2, \alpha, \boldsymbol{\delta})$ is given by

$$P(\boldsymbol{\beta}, \sigma^2, \alpha, \boldsymbol{\delta}|\boldsymbol{y}) \propto l(\boldsymbol{y}|\boldsymbol{\beta}, \sigma^2, \alpha)\, P_{\beta,\sigma^2}(\boldsymbol{\beta}, \sigma^2)\, P_{\delta|\alpha}(\boldsymbol{\delta}|\alpha)\, P_\alpha(\alpha),$$

where

$$\begin{aligned}
&P_\alpha(\alpha) = \frac{\Gamma(\gamma_1+\gamma_2)}{\Gamma(\gamma_1)\,\Gamma(\gamma_2)} \alpha^{\gamma_1-1}(1-\alpha)^{\gamma_2-1}, \text{ as } \alpha \sim \mathrm{Beta}\,(\gamma_1, \gamma_2),\\
&P_{\delta|\alpha}(\boldsymbol{\delta}|\alpha) = \alpha^{\sum \delta_i}(1-\alpha)^{n-\sum \delta_i}, \text{ as } \mathbb{P}(\delta_i = 1|\alpha) = \alpha,\\
&P_{\beta,\sigma^2}(\boldsymbol{\beta}, \sigma^2) = \frac{1}{\sigma^2},\\
&l(\boldsymbol{y}|\boldsymbol{\beta}, \sigma^2, \alpha) = \frac{1}{\sigma^n |V|^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{y}-X\boldsymbol{\beta})^{\mathrm{T}} V^{-1}(\boldsymbol{y}-X\boldsymbol{\beta})\right\} \text{ as}\\
&\qquad \boldsymbol{y}|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta} \sim \mathrm{N}_n(X\boldsymbol{\beta}, \sigma^2 V), \text{ with } V = \mathrm{diag}(1+\delta_i(k^2-1)).
\end{aligned}$$

Then $P(\boldsymbol{\beta}, \sigma^2, \alpha, \boldsymbol{\delta}|\boldsymbol{y})$

$$\propto \alpha^{\gamma_1+\sum\delta_i-1}(1-\alpha)^{\gamma_2+n-\sum\delta_i-1} \frac{1}{\sigma^{2(\frac{n}{2}+1)}|V|^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{y}-X\boldsymbol{\beta})^{\mathrm{T}} V^{-1}(\boldsymbol{y}-X\boldsymbol{\beta})\right\}.$$

We want to find the marginal posterior distribution of $\boldsymbol{\delta}$, where

$$P(\boldsymbol{\delta}|\boldsymbol{y}) = \int_{\sigma^2}\int_{\beta}\int_{\alpha} P(\boldsymbol{\beta}, \sigma^2, \alpha, \boldsymbol{\delta}|\boldsymbol{y})\,\mathrm{d}\alpha\,\mathrm{d}\boldsymbol{\beta}\,\mathrm{d}\sigma^2.$$

The marginal posterior distribution of $(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta})$, $P(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta}|\boldsymbol{y})$

$$\begin{aligned}
&= \int_{\alpha} P(\boldsymbol{\beta}, \sigma^2, \alpha, \boldsymbol{\delta}|\boldsymbol{y})\,\mathrm{d}\alpha \\
&\propto \frac{1}{\sigma^{2(\frac{n}{2}+1)}|V|^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{y}-X\boldsymbol{\beta})^{\mathrm{T}}V^{-1}(\boldsymbol{y}-X\boldsymbol{\beta})\right\} \\
&\quad \int_{\alpha} \alpha^{\gamma_1+\sum\delta_i-1}(1-\alpha)^{\gamma_2+n-\sum\delta_i-1}\mathrm{d}\alpha \\
&\propto \frac{1}{\sigma^{2(\frac{n}{2}+1)}|V|^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{y}-X\boldsymbol{\beta})^{\mathrm{T}}V^{-1}(\boldsymbol{y}-X\boldsymbol{\beta})\right\} \frac{\Gamma(\gamma_1+\sum\delta_i)\,\Gamma(\gamma_2+n-\sum\delta_i)}{\Gamma(\gamma_1+\gamma_2+n)} \\
&\quad \int_{\alpha} \frac{\Gamma(\gamma_1+\gamma_2+n)}{\Gamma(\gamma_1+\sum\delta_i)\,\Gamma(\gamma_2+n-\sum\delta_i)} \alpha^{\gamma_1+\sum\delta_i-1}(1-\alpha)^{\gamma_2+n-\sum\delta_i-1}\mathrm{d}\alpha \\
&\propto \frac{1}{\sigma^{2(\frac{n}{2}+1)}|V|^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{y}-X\boldsymbol{\beta})^{\mathrm{T}}V^{-1}(\boldsymbol{y}-X\boldsymbol{\beta})\right\} \frac{\Gamma(\gamma_1+\sum\delta_i)\,\Gamma(\gamma_2+n-\sum\delta_i)}{\Gamma(\gamma_1+\gamma_2+n)},
\end{aligned}$$

by deducing that $\alpha|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta} \sim \text{Beta}\,(\gamma_1+\sum\delta_i, \gamma_2+n-\sum\delta_i)$.

We define $a_\delta = \frac{\Gamma(\gamma_1+\sum\delta_i)\Gamma(\gamma_2+n-\sum\delta_i)}{\Gamma(\gamma_1+\gamma_2+n)}$. The marginal posterior distribution of $(\sigma^2, \boldsymbol{\delta})$, $P(\sigma^2, \boldsymbol{\delta}|\boldsymbol{y})$

$$\begin{aligned}
&= \int_{\beta} P(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta}|\boldsymbol{y})\,\mathrm{d}\boldsymbol{\beta} \\
&\propto a_\delta \frac{1}{\sigma^{2(\frac{n}{2}+1)}|V|^{1/2}} \int_{\beta} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{y}-X\boldsymbol{\beta})^{\mathrm{T}}V^{-1}(\boldsymbol{y}-X\boldsymbol{\beta})\right\}\mathrm{d}\boldsymbol{\beta} \\
&\propto a_\delta \frac{1}{\sigma^{2(\frac{n}{2}+1)}|V|^{1/2}} \\
&\quad \int_{\beta} \exp\left\{-\frac{1}{2\sigma^2}\left[(\boldsymbol{y}-X\hat{\boldsymbol{\beta}})+(X\hat{\boldsymbol{\beta}}-X\boldsymbol{\beta})\right]^{\mathrm{T}}V^{-1}\left[(\boldsymbol{y}-X\hat{\boldsymbol{\beta}})+(X\hat{\boldsymbol{\beta}}-X\boldsymbol{\beta})\right]\right\}\mathrm{d}\boldsymbol{\beta} \\
&\propto a_\delta \frac{1}{\sigma^{2(\frac{n}{2}+1)}|V|^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{y}-X\hat{\boldsymbol{\beta}})^{\mathrm{T}}V^{-1}(\boldsymbol{y}-X\hat{\boldsymbol{\beta}})\right\} \\
&\quad \int_{\beta} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})^{\mathrm{T}}(X^{\mathrm{T}}V^{-1}X)(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})\right\}\mathrm{d}\boldsymbol{\beta}
\end{aligned}$$

since $(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})^{\mathrm{T}}X^{\mathrm{T}}V^{-1}(\boldsymbol{y}-X\hat{\boldsymbol{\beta}}) = 0$, where $\hat{\boldsymbol{\beta}} = (X^{\mathrm{T}}V^{-1}X)^{-1}X^{\mathrm{T}}V^{-1}\boldsymbol{y}$.

We deduce that $\boldsymbol{\beta}|\boldsymbol{y},\sigma^2,\boldsymbol{\delta} \sim \mathrm{N}_p(\hat{\boldsymbol{\beta}},\, \sigma^2(X^{\mathrm{T}}V^{-1}X)^{-1})$. Then $P(\sigma^2,\boldsymbol{\delta}|\boldsymbol{y})$

$$
\begin{aligned}
&\propto\ a_\delta \frac{1}{\sigma^{2(\frac{n}{2}+1)}|V|^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{y}-X\hat{\boldsymbol{\beta}})^{\mathrm{T}}V^{-1}(\boldsymbol{y}-X\hat{\boldsymbol{\beta}})\right\}\left(\sigma^{2p}|(X^{\mathrm{T}}V^{-1}X)^{-1}|\right)^{1/2} \\
&\quad \int_\beta \frac{1}{\left(\sigma^{2p}|(X^{\mathrm{T}}V^{-1}X)^{-1}|\right)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})^{\mathrm{T}}(X^{\mathrm{T}}V^{-1}X)(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})\right\} \mathrm{d}\boldsymbol{\beta} \\
&\propto\ a_\delta \frac{1}{\sigma^{2(\frac{n}{2}+1)}|V|^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{y}-X\hat{\boldsymbol{\beta}})^{\mathrm{T}}V^{-1}(\boldsymbol{y}-X\hat{\boldsymbol{\beta}})\right\}\left(\sigma^{2p}|(X^{\mathrm{T}}V^{-1}X)^{-1}|\right)^{1/2} \\
&\propto\ a_\delta \left(\frac{|(X^{\mathrm{T}}V^{-1}X)^{-1}|}{|V|}\right)^{1/2} \frac{1}{\sigma^{2(\frac{n-p}{2}+1)}} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{y}-X\hat{\boldsymbol{\beta}})^{\mathrm{T}}V^{-1}(\boldsymbol{y}-X\hat{\boldsymbol{\beta}})\right\}.
\end{aligned}
$$

The marginal posterior distribution of $\boldsymbol{\delta}$, $P(\boldsymbol{\delta}|\boldsymbol{y})$

$$
\begin{aligned}
&=\ \int_{\sigma^2} P(\sigma^2,\boldsymbol{\delta}|\boldsymbol{y})\,\mathrm{d}\sigma^2 \\
&\propto\ a_\delta \left(\frac{|(X^{\mathrm{T}}V^{-1}X)^{-1}|}{|V|}\right)^{1/2} \int_{\sigma^2} \frac{1}{\sigma^{2(\frac{n-p}{2}+1)}} \exp\left\{-\frac{1}{2}(\boldsymbol{y}-X\hat{\boldsymbol{\beta}})^{\mathrm{T}}V^{-1}(\boldsymbol{y}-X\hat{\boldsymbol{\beta}})\frac{1}{\sigma^2}\right\} \mathrm{d}\sigma^2 \\
&\propto\ a_\delta \left(\frac{|(X^{\mathrm{T}}V^{-1}X)^{-1}|}{|V|}\right)^{1/2} \frac{\Gamma\left(\frac{n-p}{2}\right)}{\left[\frac{(y-X\hat{\beta})^{\mathrm{T}}V^{-1}(y-X\hat{\beta})}{2}\right]^{\frac{(n-p)}{2}}} \\
&\quad \int_{\sigma^2} \frac{\left[\frac{(y-X\hat{\beta})^{\mathrm{T}}V^{-1}(y-X\hat{\beta})}{2}\right]^{\frac{(n-p)}{2}}}{\Gamma\left(\frac{n-p}{2}\right)} \frac{1}{\sigma^{2(\frac{n-p}{2}+1)}} \exp\left\{-\frac{1}{2}(\boldsymbol{y}-X\hat{\boldsymbol{\beta}})^{\mathrm{T}}V^{-1}(\boldsymbol{y}-X\hat{\boldsymbol{\beta}})\frac{1}{\sigma^2}\right\} \mathrm{d}\sigma^2 \\
&\propto\ a_\delta \left(\frac{|(X^{\mathrm{T}}V^{-1}X)^{-1}|}{|V|}\right)^{1/2} \frac{\Gamma\left(\frac{n-p}{2}\right)}{\left[\frac{(y-X\hat{\beta})^{\mathrm{T}}V^{-1}(y-X\hat{\beta})}{2}\right]^{\frac{(n-p)}{2}}},
\end{aligned}
$$

by deducing that $\sigma^2|\boldsymbol{y} \sim \mathrm{Inv}-\mathrm{Gamma}\left(\frac{n-p}{2},\ \frac{(y-X\hat{\beta})^{\mathrm{T}}V^{-1}(y-X\hat{\beta})}{2}\right)$.

Therefore, the marginal posterior distribution of $\boldsymbol{\delta}$ is given by

$$
P(\boldsymbol{\delta}|\boldsymbol{y}) \propto \left(\frac{|(X^{\mathrm{T}}V^{-1}X)^{-1}|}{|V|}\right)^{1/2} \frac{\Gamma(\gamma_1+\sum\delta_i)\,\Gamma(\gamma_2+n-\sum\delta_i)}{\left[(\boldsymbol{y}-X\hat{\boldsymbol{\beta}})^{\mathrm{T}}V^{-1}(\boldsymbol{y}-X\hat{\boldsymbol{\beta}})\right]^{\frac{(n-p)}{2}}}.
$$

Appendix D

Data set: Stack loss

This data set is a group of data from a plant for the oxidation of ammonia to nitric acid. 21 diary observations were collected for three explanatory variables X_1, X_2, X_3 and one response variable Y. X_1 is the flow of the air to the plant, X_2 is the cooling water inlet temperature, X_3 is the acid concentration while Y is 10 times the percentage of the ingoing ammonia that is lost. More information about this data set can be found in Daniel & Wood (1980).

Observation number	Y	X_1	X_2	X_3
1	42	80	27	89
2	37	80	27	88
3	37	75	25	90
4	28	62	24	87
5	18	62	22	87
6	18	62	23	87
7	19	62	24	93
8	20	62	24	93
9	15	58	23	87
10	14	58	18	80
11	14	58	18	89
12	13	58	17	88
13	11	58	18	82
14	12	58	19	93
15	8	50	18	89
16	7	50	18	86
17	8	50	19	72
18	8	50	19	79
19	9	50	20	80
20	15	56	20	82
21	15	70	20	91

Table D.1: Values of the stack loss data

Appendix E

Data set: Southern Oscillation Index

This data set consists of 540 monthly observations of the southern oscillation index during 1950–1995, where x is the "difference of the departure from the long-term monthly mean sea-level pressures at Tahiti in the South Pacific and Darwin in Northern Australia".

x (by row)																
2.7	1.0	-1.3	-1.1	-1.7	-0.5	-2.3	-1.2	-2.1	-2.3	-1.6	-1.6	-2.0	-1.8	0.0	-0.9
1.0	0.8	0.7	-0.7	-0.4	0.3	-0.3	-2.6	0.4	-1.6	-1.4	-0.1	-3.6	-0.5	-0.2	-3.1
-2.4	-0.3	-0.7	-1.1	1.0	-1.2	-0.5	0.6	0.4	-0.5	0.4	1.3	0.3	0.1	0.2	2.4
-1.1	2.9	0.1	-0.8	1.4	1.7	2.7	2.0	2.5	2.5	2.2	1.7	2.3	2.4	1.5	1.2
2.2	1.3	1.7	1.5	0.0	3.0	0.1	1.7	1.0	-0.8	-0.6	0.0	-1.7	-0.4	0.1	-1.7
-1.8	-0.3	-2.0	-0.9	-3.8	-1.6	-0.5	0.2	-1.5	-0.2	0.4	1.0	-0.7	-0.4	-1.0	-1.4
-2.0	-3.2	1.4	0.3	0.4	-0.9	-0.8	-0.9	-0.0	0.5	1.6	1.4	0.0	-0.4	0.9	1.0
0.5	-0.5	0.6	0.8	1.1	-0.2	0.9	1.3	-0.7	1.1	-4.5	1.1	0.2	-0.5	0.2	-0.4
0.1	-1.1	0.9	2.6	3.6	-1.1	-0.7	-0.1	1.6	0.6	-0.2	0.6	0.8	1.4	0.4	0.1
1.8	0.6	1.1	1.0	0.2	-1.6	-0.5	-0.8	-1.1	-2.7	-1.7	-2.6	-0.9	-0.5	1.1	1.7
-0.1	0.7	0.7	2.2	2.3	2.0	0.1	-0.9	-1.1	0.1	0.4	-1.4	-0.2	-1.7	-3.6	-2.0
-2.5	-2.0	-2.9	0.0	-2.8	-1.1	-2.8	-0.8	-1.1	-0.0	-0.2	0.5	-0.4	-0.6	-0.2	-1.0
3.0	2.6	1.3	-0.4	-0.5	0.6	0.0	0.7	1.0	-0.3	-1.0	-1.4	0.6	1.8	-0.8	-0.4
1.9	1.4	1.0	-0.2	-0.5	-0.5	-0.7	0.1	-3.2	-1.8	-0.2	-1.1	-0.9	-0.3	-1.2	-0.9
-2.0	-2.2	-0.3	0.5	-2.3	-2.7	0.1	-0.6	0.2	1.2	-1.0	0.4	2.1	1.5	2.8	3.3
0.4	3.1	3.4	2.8	1.2	0.2	0.1	2.2	2.7	2.8	0.9	0.1	0.6	1.4	0.1	-0.6
-3.4	-1.8	-3.1	-1.6	-2.6	-2.0	-0.7	-2.6	-0.8	-3.2	0.3	-0.3	0.4	1.3	0.9	1.7
2.3	1.0	4.7	3.2	4.3	3.2	3.6	1.4	1.4	0.1	1.9	0.8	2.0	1.3	-0.5	-0.0
-1.3	1.0	1.9	1.7	0.8	1.8	3.4	3.1	3.9	2.8	2.1	3.7	2.4	2.6	2.2	0.2
0.3	-0.2	-1.9	-2.2	-2.2	0.4	1.1	-1.0	-1.1	1.7	-2.1	-1.3	-1.4	-2.5	-2.5	-2.2
-1.6	-2.3	-2.5	-2.3	-0.7	-5.7	-1.3	-1.0	2.1	0.5	0.7	0.0	0.1	-1.2	-0.2	-0.4
-1.1	1.3	-0.8	-0.7	0.5	0.6	2.2	-1.0	0.2	-0.6	-1.0	-1.6	0.5	-0.0	-2.0	-1.7
-0.5	-0.7	-0.4	0.0	-0.9	-0.4	-0.8	-0.5	0.4	-1.0	-3.4	-0.7	1.1	1.7	1.2	0.6
0.6	-1.2	0.1	0.7	2.2	-0.1	0.1	-0.3	-1.1	-2.6	-3.2	-4.0	-3.3	-3.6	-5.1	-4.6
-6.9	-7.6	-5.6	-2.2	0.7	-0.5	-1.3	-0.3	1.7	0.4	-0.3	-0.2	0.2	0.9	-1.5	0.3
-0.1	-1.3	0.1	0.1	0.2	-1.0	0.4	-0.7	-0.7	1.7	0.3	1.7	0.3	-1.5	-0.4	1.1
-0.1	-1.2	-0.5	0.2	1.5	-2.7	-0.1	0.1	-0.9	1.1	0.2	-1.6	-1.0	0.9	-2.5	-3.0
-1.5	-3.1	-3.3	-3.0	-2.8	-2.8	-2.8	-2.5	-1.9	-1.1	-0.2	-1.2	-0.3	-1.4	0.1	-0.1
1.3	-0.4	1.7	2.2	3.4	2.2	3.0	2.1	2.7	1.8	1.0	2.6	1.9	0.8	1.4	-1.3
0.9	1.0	-0.6	-1.2	-0.4	-3.9	-1.9	-0.1	1.8	-0.1	0.8	-1.0	-1.3	0.1	-1.1	-0.7
1.0	-0.1	-2.2	-1.7	-2.4	-0.9	-0.2	-1.4	-2.9	-2.4	-1.4	-3.7	-5.6	-2.3	-4.8	-2.3
0.1	-1.9	-1.3	-0.0	0.0	-3.2	-1.4	-1.4	-2.0	-2.1	-1.8	-2.6	-1.0	-2.2	-1.8	-2.4
-1.3	-2.5	-0.3	0.1	-0.5	-0.1	-2.2	-2.9	-1.7	-1.5	-2.9	-3.0	-3.0	-2.6	-1.2	-2.6
-1.0	-0.8	0.4	-1.8	-1.2	-0.4	0.6	-0.1	0.5	-0.5	-0.1	-1.3				

Table E.1: Values of the Southern Oscillation Index data

Bibliography

Al-Awadhi, F., Hurn, M. & Jennison, C. (2004), 'Statistical image analysis for a confocal microscopy two-dimensional section of cartilage growth', *Applied Statistics, Journal of the Royal Statistical Society, Series C* **53**, 31–49.

Andrieu, C. & Moulines, E. (2006), 'On the ergodic properties of some adaptive MCMC algorithms', *Annals of Applied Probability* **16**(3), 1462–1505.

Atchade, Y. F. & Rosenthal, J. S. (2005), 'On adaptive Markov chain Monte Carlo algorithms', *Bernoulli* **11**, 815–828.

Avriel, M. (1976), *Nonlinear programming: analysis and methods*, Prentice-Hall, USA.

Besag, J. (1974), 'Spatial interaction and the statistical analysis of lattice systems', *Journal of the Royal Statistical Society: Series B* **36**, 192–236.

Besag, J., York, J. & Mollie, A. (1991), 'Bayesian image restoration, with two applications in spatial statistics', *Annals of the Institute of Statistical Mathematics* **43**, 1–59.

Box, G. E. P. & Tiao, G. C. (1968), 'A Bayesian approach to some outlier problems', *Biometrika* **55**(1), 119–129.

Brooks, S. P., Giudici, P. & Roberts, G. O. (2003), 'Efficient construction of reversible jump Markov chain Monte Carlo proposal distributions', *Journal of the Royal Statistical Society: Series B* **65**(1), 3–55.

Daniel, C. & Wood, F. S. (1980), *Fitting equations to data*, 2nd edn, Wiley, Chichester.

Everitt, B. S., Landau, S. & Leese, M. (2001), *Cluster analysis*, 4th edn, Arnold, London.

Franconi, L. & Jennison, C. (1993), A comparison of genetic algorithms and simulated annealing in an application to statistical image reconstruction, Statistics Research Report 93:09, University of Bath.

Gasemyr, J. (2003), 'On an adaptive version of the Metropolis–hastings algorithm with independent proposal distribution', *Scandinavian Journal of Statistics* **30**(1), 159–173.

Gelman, A. (1996), Inference and monitoring convergence, *in* W. R. Gilks, S. Richardson & D. Spiegelhalter, eds, 'Markov Chain Monte Carlo in Practice', Chapman & Hall, London, pp. 131–143.

Geman, S. & Geman, D. (1984), 'Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**, 721–741.

Geyer, C. J. (1991), Markov chain Monte Carlo maximum likelihood, *in* E. M. Keramidas, ed., 'Computer Science and Statistics: Proceedings of the 23rd Symposium on the Interface', Interface Foundation, Fairfax Station, pp. 156–163.

Geyer, C. J. & Thompson, E. A. (1995), 'Annealing Markov chain Monte Carlo with applications to ancestral inference', *Journal of the American Statistical Association* **90**, 909–920.

Gilks, W., Roberts, G. & Sahu, S. (1998), 'Adaptive Markov chain Monte Carlo through regeneration', *Journal of the American Statistical Association* **93**, 1045–1054.

Glasbey, C. A. & Horgan, G. W. (1995), *Image analysis for the biological sciences*, John Wiley & Sons, Chicester.

Gray, A. J. (1994), 'Simulating posterior Gibbs distributions: a comparison of the Swendsen–Wang and Gibbs sampler methods', *Statistics and Computing* **4**, 189–201.

Green, P. J. (1995), 'Reversible jump Markov chain Monte Carlo computation and Bayesian model determination', *Biometrika* **82**(4), 711–732.

Green, P. J. (2003), Trans-dimensional Markov chain Monte Carlo, *in* P. J. Green, N. L. Hjort & S. Richardson, eds, 'Highly Structured Stochastic Systems', Oxford University Press, Oxford, pp. 178–198.

Green, P. J. & Han, X.-L. (1992), Metropolis methods, Gaussian proposals and antithetic variables, *in* P. Barone, A. Frigressi & M. Piccioni, eds, 'Stochastic Models, Statistical Methods and Algorithms in Image Analysis', Vol. 74 of *Lecture Notes in Statistics*, Springer, Berlin, pp. 142–164.

Green, P. J. & Richardson, S. (1997), 'On Bayesian analysis of mixtures with an unknown number of components', *Journal of the Royal Statistical Society: Series B* **59**(4), 731–792.

Grimmett, G. R. & Stirzaker, D. R. (2001), *Probability and random processes*, 3rd edn, Oxford University Press, Oxford.

Haario, H., Saksman, E. & Tamminen, J. (2001), 'An adaptive Metropolis algorithm', *Bernoulli* **7**(2), 223–242.

Haario, H., Saksman, E. & Tamminen, J. (2005), 'Componentwise adaptation for high dimensional MCMC', *Computational Statistics* **20**, 265–273.

Hajek, B. (1988), 'Cooling schedules for optimal annealing', *Mathematics of Operational Research* **13**, 311–329.

Hastings, W. K. (1970), 'Monte Carlo sampling methods using Markov chains and their applications', *Biometrika* **57**, 97–109.

Hurn, M., Justel, A. & Robert, C. P. (2003), 'Estimating mixtures of regressions', *Journal of Computational and Graphical Statistics* **12**(1), 55–79.

Jennison, C. (1993), 'Discussion on the meeting on the Gibbs sampler and other Markov chain Monte Carlo methods', *Journal of the Royal Statistical Society: Series B* **55**, 54–56.

Justel, A. & Peña, D. (1996), 'Gibbs sampling will fail in outlier problems with strong masking', *Journal of Computational and Graphical Statistics* **5**(2), 176–189.

Justel, A. & Peña, D. (2001), 'Bayesian unmasking in linear models', *Computational Statistics & Data Analysis* **36**, 69–84.

Kirkpatrick, S., Gelatt, Jr., C. D. & Vecchi, M. P. (1983), 'Optimization by simulated annealing', *Science* **220**, 671–680.

Marinari, E. & Parisi, G. (1992), 'Simulated tempering: a new Monte Carlo scheme', *Europhysics Letters* **19**(6), 451–458.

Matheron, G. (1975), *Random sets and integral geometry*, John Wiley & Sons, New York.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. (1953), 'Equations of state calculations by fast computing machine', *Journal of Chemical Physics* **21**, 1087–1091.

Mohr, D. L. (2007), 'Bayesian identification of clustered outliers in multiple regression', *Computational Statistics & Data Analysis* **51**, 3955–3967.

Mykland, P., Tierney, L. & Yu, B. (1995), 'Regeneration in Markov chain samplers', *Journal of the American Statistical Association* **90**, 233–241.

Neal, R. M. (1996), 'Sampling from multimodal distributions using tempered transitions', *Statistics and Computing* **6**, 353–366.

Roberts, G. O. (1996), Markov chain concepts related to sampling algorithms, *in* W. R. Gilks, S. Richardson & D. Spiegelhalter, eds, 'Markov Chain Monte Carlo in Practice', Chapman & Hall, London, pp. 45–58.

Rousseeuw, P. J. & Leroy, A. M. (1987), *Robust regression and outlier detection*, John Wiley & Sons, USA.

Sahu, S. K. & Zhigljavsky, A. A. (2003), 'Self–regenerative Markov chain Monte Carlo with adaptation', *Bernoulli* **9**, 395–422.

Serra, J. (1982), *Image analysis and mathematical morphology*, Academic Press, London.

Sharp, R. M. (2003), An approximate alternative to perfect simulation, PhD thesis, Department of Mathematical Sciences, University of Bath.

Swenden, R. H. & Wang, J. S. (1987), 'Nonuniversal critical dynamics in Monte Carlo simulation', *Physics Review Letters* **58**, 86–88.

Tierney, L. (1996), Introduction to general state-space Markov chain theory, *in* W. R. Gilks, S. Richardson & D. Spiegelhalter, eds, 'Markov Chain Monte Carlo in Practice', Chapman & Hall, London, pp. 59–74.

Tjelmeland, H. & Hegstad, B. K. (2001), 'Mode jumping proposals in MCMC', *Scandinavian Journal of Statistics* **28**, 205–223.

Verdinelli, I. & Wasserman, L. (1991), 'Bayesian analysis of outlier problems using the Gibbs sampler', *Statistics and Computing* **1**, 105–117.